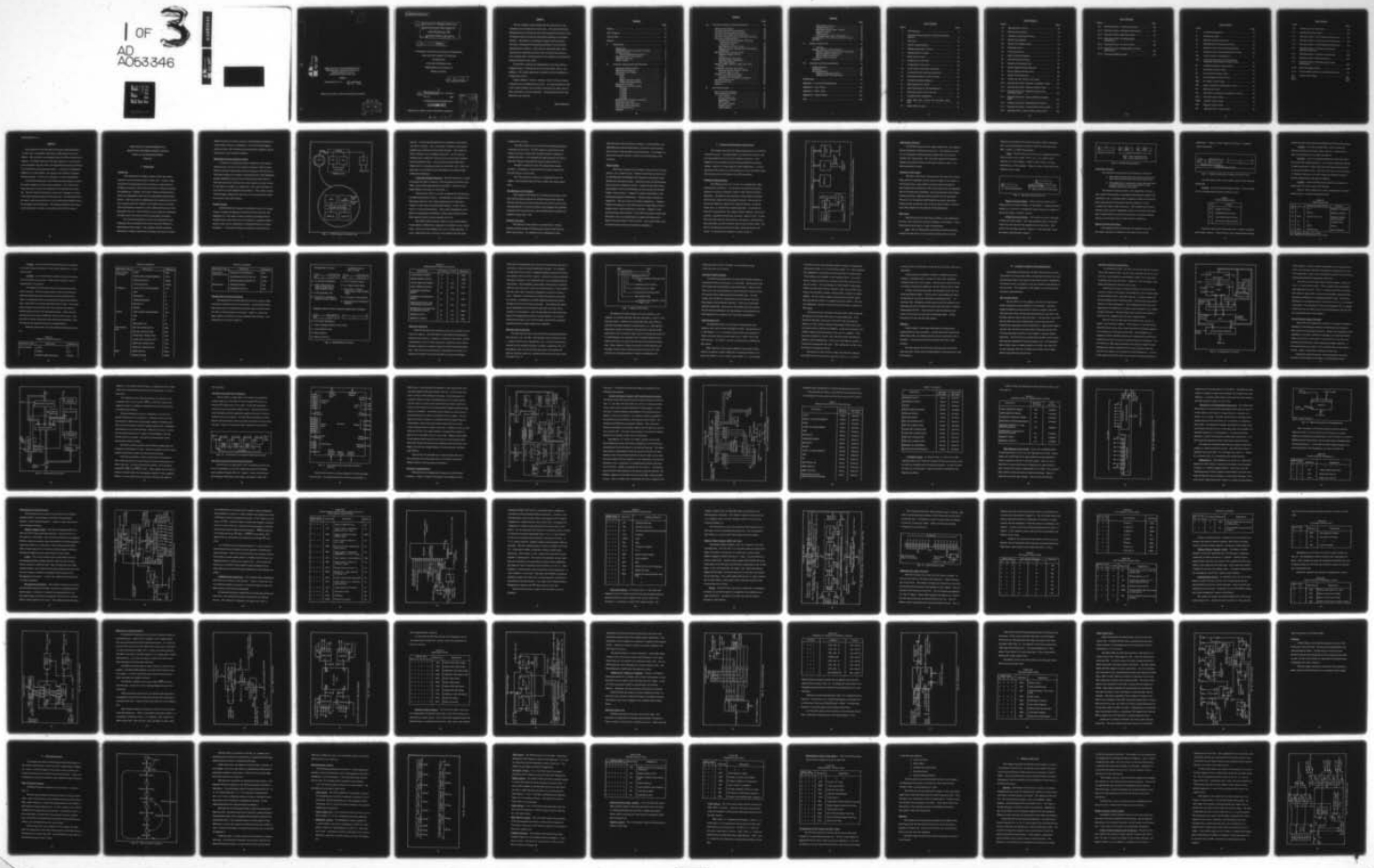
AD-A053 346 AIR FORCE INST OF TECH WRIGHT-PATTERSON AFB OHIO SCH--ETC F/G 9/2
EMULATION OF THE PROCESSOR FOR DISTRIBUTED PROCESSOR/MEMORY SYS--ETC(U)
DEC 77 E MUHAMMAD

UNCLASSIFIED AFIT/GE/EE/77-31 NL

1 OF 3
AD A053346



EMULATION OF THE PROCESSOR FOR DISTRIBUTED PROCESSOR/MEMORY SYSTEM USING Am 2900 MICROPROCESSOR CHIP SET

THESIS

AFIT/GE/EE/77-31 Ejaz Muhammad
Flt. Lt. PAF

14 AFIT/GE/EE/77-31

6 EMULATION OF THE PROCESSOR FOR DISTRIBUTED PROCESSOR/MEMORY SYSTEM USING Am 2900 MICROPROCESSOR CHIP SET,

9 Master's THESIS,

Presented to the Faculty of the School of Engineering

of the Air Force Institute of Technology

Air University

in Partial Fulfillment of the

Requirements for the Degree of

Master of Science

12 240p.

by

10 Ejaz Muhammad, B.E. (Avionics)

Flt Lt PAF

Graduate Electrical Engineering

11 December 1977

Approved for public release; distribution unlimited

012 225

Preface

The art of digital system design with microprocessors, has acquired a new momentum in recent years. The advent of Bit Slice Microprocessors, like the Am 2900, has become an attractive source of enhanced speed of operation and greater flexibility in the digital systems. This thesis is an attempt to design a special purpose processor utilizing the aforementioned attributes of the Am 2900 microprocessor chip set. I have tried to outline the steps which represent the sequential procedure used to develop the processor. I am confident that a working model can be realized by hardwiring the design presented in this report.

I would like to express my indebtedness to my thesis advisor, Captain James B. Peterson for his very timely help, advice, and guidance. His valued suggestions sustained in me the confidence to complete this report.

I wish to thank Dr. Mark T. Michael of the Air Force Avionics Laboratory for providing the thesis topic. My special thanks are due to Mr. Deiter Schiller of the Avionics Laboratory for many hours of help, discussion, and encouragement. I greatly appreciate his keen interest in my research.

Ejaz Muhammad

Contents

Contents

Contents

List of Figures

List of Figures

List of Figures

List of Tables

List of Tables

Abstract

The processor for the Distributed Processor (DP/M) System is a 16-bit, two's complement, fixed point, eight register file architecture. This processor was designed using Am 2900 microprocessor chip set. The design used four CPU chips (Am 2901), one microprogram controller chip (Am 2910), and eight PROM chips (Intel 3604A-2) to implement the microprogram memory. A number of multiplexers, registers, tri-state buffers, and counters were utilized to augment the basic design. A micro-level "Monitor" was also implemented.

A 64-bit microinstruction word format was determined to provide the control signals for the processor hardware. Flow charts were drawn and micro-codes were tabulated for the specified instruction set. The flow charts and the micro-codes were arranged to conform to the state transition diagram of the processor. The Am 2900 microprocessor chip set was found to be a very powerful and flexible source for emulating the DP/M System. The design presented in this report can be hardwired to realize a Lab model of the DP/M Processor.

EMULATION OF THE PROCESSOR FOR DISTRIBUTED PROCESSOR/MEMORY SYSTEM USING Am 2900 MICROPROCESSOR CHIP SET

I. Introduction

Background

The requirement for highly complex avionic and control systems in an aircraft has grown in recent years. Today, a high performance and high speed aircraft requires a large amount of in-flight processing of status and control data for effective mission accomplishment. Initially, the avionic systems performing these tasks were independent units and were implemented using analogue devices. With the advent of multifunction and sophisticated aircraft, more reliable and faster methods were needed to process and provide the information required to achieve optimal performance when flying those aircraft. This resulted in new concepts for integrating the flight control and avionic systems using digital techniques.

The recent developments in large scale integrated circuit technology and minicomputers have led to many new methods for integrating aircraft avionics. The computer systems presently operational are highly centralized in that they rely upon one central

digital processor to receive, process, and distribute information to various units, sensors, and displays. In case the central digital processor fails, the information processing stops altogether and the mission has to be drastically modified.

Distributed Processor/Memory System

In order to overcome the serious limitations of the centralized processor, the Air Force Avionics Laboratory (AFAL) formulated the concept of Distributed Processor/Memory (DP/M) System. This system would use a number of microprocessors with independent storage capability to process the raw data from aircraft sensors. Each microprocessor would process some specific raw data, format it, and make it available on a "global bus" to all other microprocessors and the aircraft equipment that needed it. This system concept would be highly reliable since the loss of one microprocessor would not result in total system failure.

Overall Concept

The DP/M System Concept is essentially the use of a varying number of simple homogeneous processor/memory elements (PE) applicable to a wide range of avionic system processing problems. Architecturally, these PEs can be used either as standalone uniprocessors or they can be configured in a distributed network as shown in Figure 1. Two levels of busing are needed to interconnect the

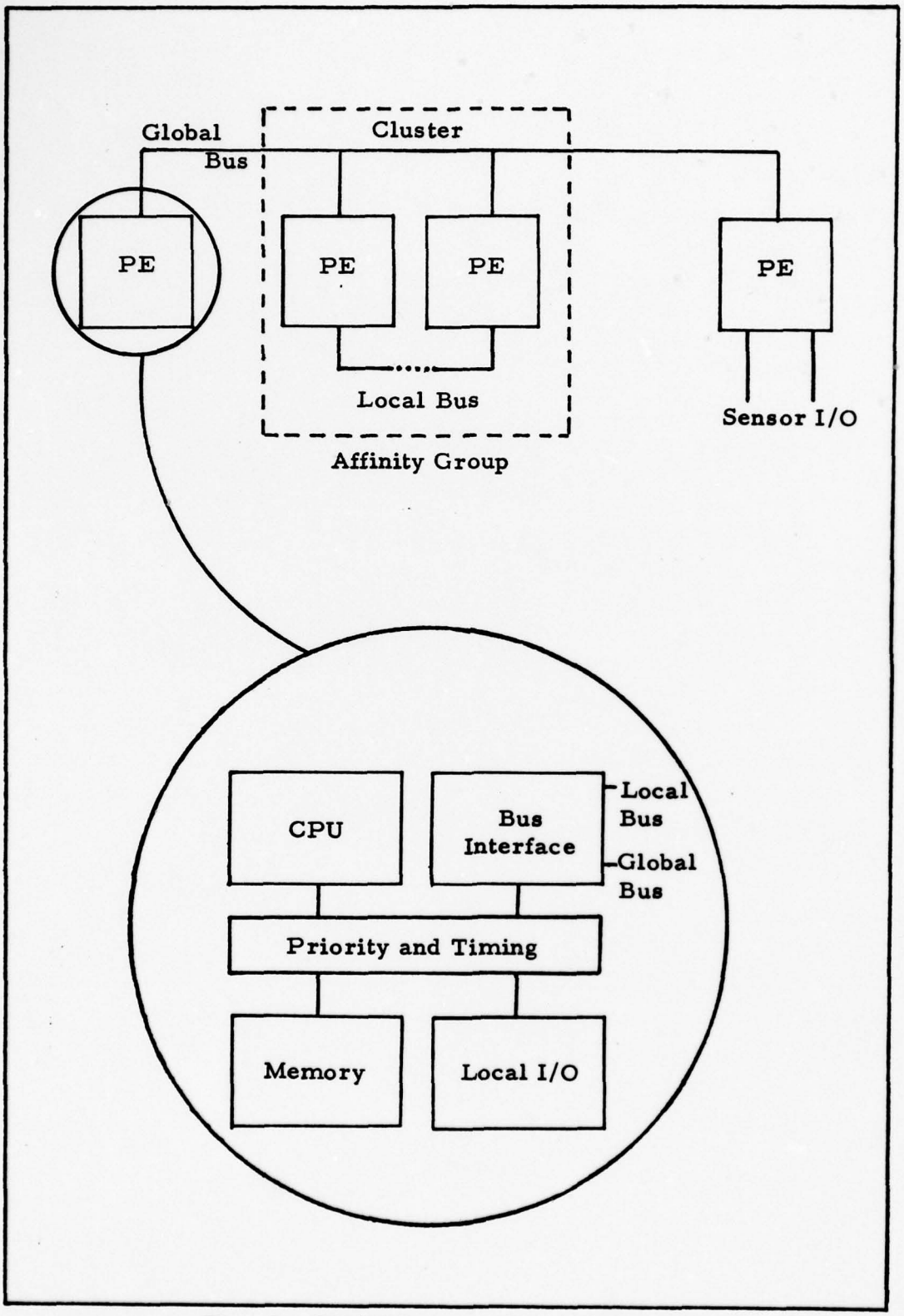


Fig. 1. DP/M System (From Ref 1:14)

network. A dual-redundant global bus is required to interconnect each PE in a system. Also, a local bus is needed to interconnect multiple PEs clustered to form a given function. This cluster of PEs is referred to as an Affinity Group (AG). An AG would be required when a single PE could not process all the data available from a particular sensor. The local bus interface allows PEs within an AG to communicate with each other (Ref 1:13). Thus, the basic idea is to decentralize the information processing through distributed intelligence.

Processing Element Structure. Each PE consists of a central processing unit (CPU), memory, local and global bus interface unit (BIU), and an input/output interface unit (IOIU). These four functional modules are shown in a box in Fig. 1.

The processor CPU is a 16-bit, 8-register file microprocessor with microprogram control. Functionally, it can address up to 65K words; however, most avionic applications are expected to require 4 to 8K words of program/data storage. It uses a set of forty 16- and 32-bit instructions. The primary emphasis has been placed on maximizing the efficiency of those high-frequency operations typically found in avionic software (Ref 1:16).

The DP/M system memory module is an 8K words Random Access Memory (RAM) with a single port accessible to users via the I-Bus. The access time ranges from 1/3 to 1/2 microseconds. It uses a single parity bit per-word to check whether the data being

transferred is correct.

The BIU provides the two levels of bus interfacing required: one global and one local. The BIU supports a distributed round-robin bus protocol scheme with message broadcast capability to multiple PEs/AGs. It can identify the input messages and vectors them into software-selectable PE memory buffers (Ref 1:17).

The IOIU provides the basic PE external data transfer facilities. Essentially, it controls the data transfer between the aircraft sensors and the I-Bus.

The I-Bus structure consists of a standardized set of 80 signals, which facilitate intra-PE data transfer and control operations.

The Statement of the Problem

The purpose of this thesis is to emulate the processor of the PE of DP/M system using the Am 2900 microprocessor chip set. The Am 2900 chip set has been selected by the AFAL in view of the recommendations of a previous thesis which attempted the emulation using Intel 3000 microprocessor and concluded that it resulted in an inflexible design (Ref 5:103).

Objective and Scope

The objective of this study is to provide AFAL with the detailed workable design of the processor using Am 2900 chip set. This would require: (1) a detailed circuit configuration using

microprocessor and the external hardware, (2) the definition of an appropriate microinstruction format, (3) the flow charts and the micro-codes for the specified macroinstructions, (4) the design of a micro-level monitor facility to ensure correct functioning of the processor.

Thesis Outline

This report contains the description of the processor development in a chronological order. Chapter II elaborates on the processor specifications and highlights the functional requirements which motivate the subsequent design. Appendix A gives the specifications of the processor furnished by AFAL. Chapter III describes important members of the Am 2900 family and then presents the detailed design of the processor. Chapter IV contains a description of the microprogramming considerations. The flow charts are given in Appendix B. The micro-codes are listed in Appendix C. Chapter V outlines the design of a micro-level monitor to help display various contents while a microinstruction is executed. Chapter VI summarizes the processor design. It also offers conclusion of this study and makes a few recommendations to simplify the design and to improve the performance of the processor. The control fields of the microinstruction format are tabulated in Appendix D.

II. Processor Requirements Specifications

This chapter describes the design specifications for the DP/M system processor. The specifications are functional in nature, and have been defined by the Air Force Avionics Laboratory. The emphasis is on achieving a standard reprogrammable processing element suitable for a wide range of avionic applications. It is also desired that the processor be fabricatable as a low cost module using the microprocessor technology in the 1980 time-frame.

Functional Characteristics

The DP/M processor is a 16 bit, two's complement, eight register-file architecture. The processor provides the necessary functions to perform arithmetic, logical, shift, and data transfer operations. Operands for the execution of instructions are obtainable from the register file and program memory, and results are placed into either the register file, program memory, processor status word, or transferred as input/output data. To provide for inter-PE communication, like transfer of data, address, and various controls, a parallel asynchronous structure, called "I-Bus," is used. The I-Bus interconnects the processor, the memory, the Bus Interface Unit (BIU), and the I/O device control unit in a daisy chain. The BIU has the highest priority in the chain, and the processor, the lowest. A functional block diagram is shown in Fig. 2.

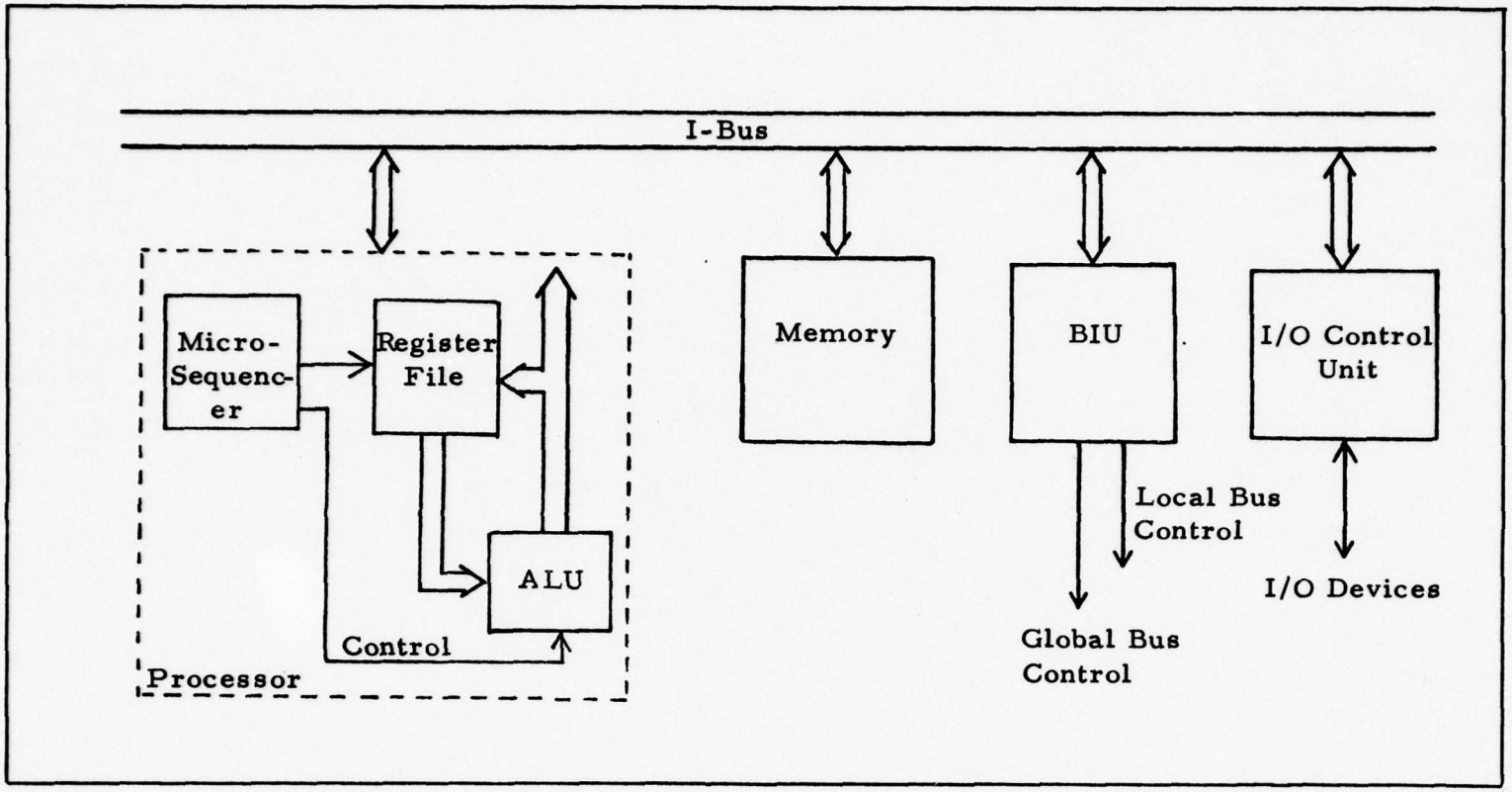


Fig. 2. Functional Block Diagram of DP/M Processing Element

Addressable Registers

The DP/M processor has an eight-register file, the registers being sequentially numbered from R_0 to R_7. The registers R_6 and R_7 are designated as Interrupt Stack Pointed (ISP) and Program Counter (PC) respectively. The rest of the registers are general purpose accumulators which can be used as indices, bases, or to hold intermediate results.

Processor Word Length

The basic word size for the processor has been set at 16 bits. This choice is compatible with the accuracy range (10 to 12 bits) of the I/O data from a large number of avionic sensors/actuators. Most numerical calculations for this sensor data can be accomplished with 16-bit fixed-point arithmetic, with some instances like multiplication and division requiring double precision (32-bit) operations. The use of a 16-bit address field within the processor instruction format permits an address reach of 64K words of memory, which is more than adequate for the expected DP/M applications.

Data Types

The DP/M processor data types available to the programmer are bits, bytes (characters and short integers), and integers. Associated with each data type is a class of instructions.

Bits. Bits are addressed by specifying the whole word that contains the desired bit or bits and then specifying the bit or bits

within the word either with a mask and using an AND or OR operation, or by specifying the bit position within the word and using a SET, CLEAR, or TEST BIT operation.

Bytes. The DP/M byte is an 8-bit two's complement number which can take on the values -128 to +127. It is used to hold a character or a short integer. Before a byte is used, it is sign extended into a 16-bit quantity. Figure 3, below, shows byte representation and its usage.

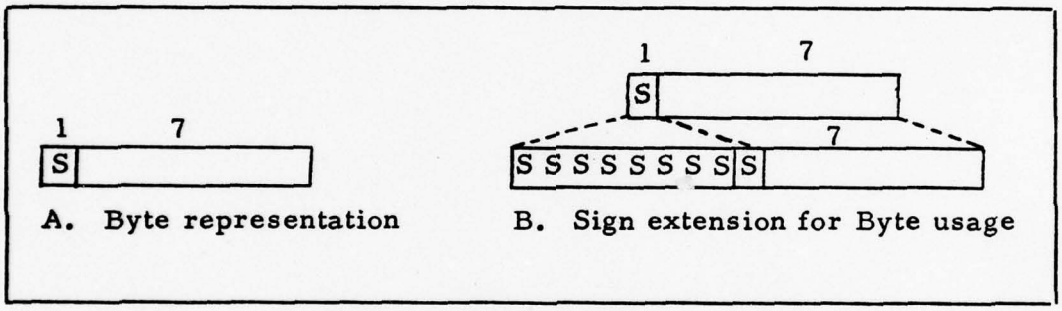


Fig. 3. Data Byte (From Ref 1:113)

Single Precision Number. Whole words are used to store single precision numbers (integer or fixed point). A single precision number is a 16-bit two's complement number which as an integer can take on the values -32,768 to +32,767.

Double Precision Number. Two words are used to represent a double precision number. The most significant part is contained in the first word and the least significant in the last word. Both words carry the same sign-bit. Figure 4, on the next page, shows the number representation schemes.

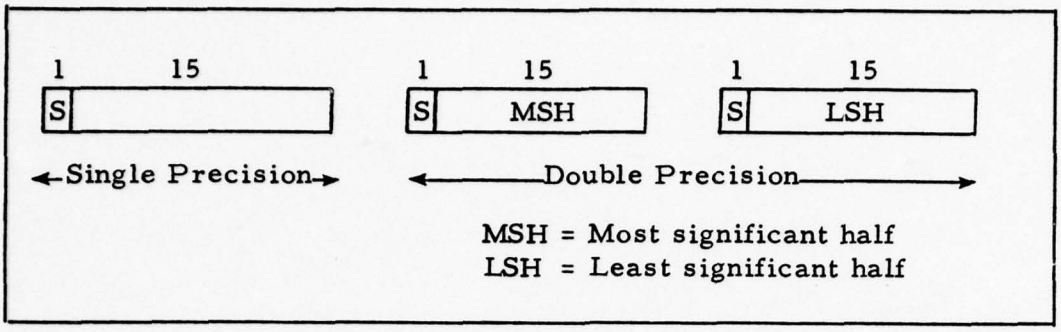


Fig. 4. Number Representation

Instruction Formats

Two instruction formats have been defined for the DP/M.

a. The standard instruction format which provides both short (one-word) and long (two-word) instructions.

b. The extended short instruction format which provides a limited number of short instructions for high frequency/efficiency operations.

The standard format provides a full complement of instructions with the desired types of operands determining the length of the instruction word. Operands found in registers require short instructions while operands requiring full memory address or 16-bit constant data require long instructions. The extended short format provides small constant (Immediate) values to be used for either data and/or displacements in one short instruction to minimize program memory and execution time for high frequency operations.

Standard Format Instructions

The standard format instructions are intended to provide a full range of operations suitable for real-time avionic system

applications. Figure 5, below, depicts the fields in a standard format instruction.

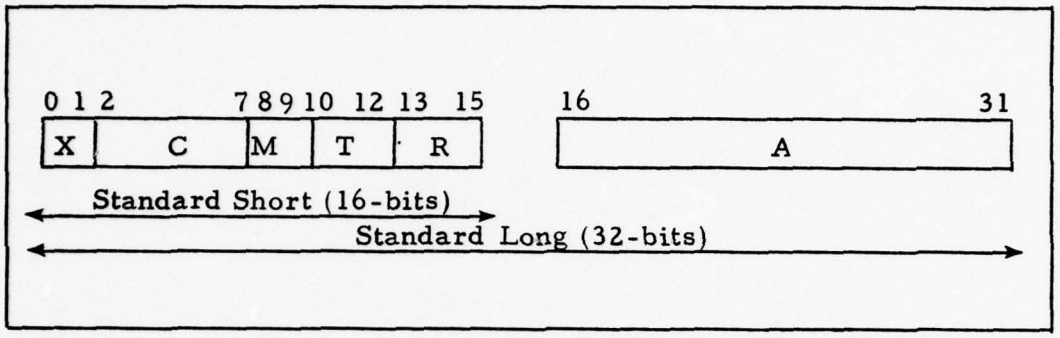


Fig. 5. Standard Instruction Format (From Ref 1:118)

The description of the various fields is given in the following paragraphs.

X-Field. A two-bit format designator field. Various values of "X" are interpreted as shown in Table I, below.

Table I
X-Field Interpretation

X	Format
00	Standard Format
01	Extended Short Format
10	Load Direct Short Format
11	Store Direct Short Format

Load/store direct short instructions are a subset of extended short format category. Thus, X-field is used to differentiate between

standard format instructions and extended short format instructions.

C-Field. A six-bit command field, used to specify the desired operation such as load, store, add, etc. These six bits allow the specification of 64 unique operations.

M-Field. A two-bit operand modification field which specifies a particular addressing mode. It is used in conjunction with the T-field and possibly A-field to determine the derived operand (DO) or the derived address (DA). The term "DA" has the same meaning as "Effective Address," and "DO" means the contents of memory location represented by the effective address.

T-Field. A three-bit field which specifies the register or index/base to be used as part of the operand.

Table II, below, defines the various combinations of M and T fields to point to a specific addressing mode.

Table II
Addressing Modes

M	T	Derived Operand/Address	Function
00	t	(t)/-	Register to Register
01	t	[(t)]/(t)	Register Indirect
10	t(t≠7)	[(t)]/(t), (t)+1→(t)	Register Indirect Autoincrement
10	t=7	A/-	Constant Data
11	t=0	[A]/A	Direct
11	t(t≠0)	[(t)]+A]/(t)+A	Direct Indexed

(Y) = Register pointed to by Y-field.
[Z] = Memory location pointed to by Z-field.

R-Field. A three-bit field which specifies the accumulator to be used during the operation or the branch condition for a conditional branch.

A-Field. A 16-bit field which specifies the next instruction word in the instruction stream. When used it contains a word of constant data or an address.

The standard format instructions are broadly categorized into two groups. The first group consists of those instructions in which the addressing modes determine the derived operand (DO) which, in conjunction with a second operand (usually in the register file), would effect an operation specified by the C-field. The second group comprises those instructions which need the calculation of derived address (DA) for the specified operation. Thus, the first group instructions need an additional "Memory Read" cycle. This point will be further elaborated in the discussion pertaining to effective address calculation and processor implementation.

Table III, given below, lists the standard format instruction set.

Table III
Standard Format Instructions

Instruction Type	Instruction	Mnemonic
Data Transfer	LOAD	L
	STORE	ST
	STORE THROUGH MASK	STTM

Table III (continued)

Instruction Type	Instruction	Mnemonic
Data Transfer (continued)	PUSH	PSH
	MOVE AND AUTOINCREMENT	MVA
	PUSH MULTIPLE	PSHM
	POP MULTIPLE	POPM
Arithmetic	LOAD TWO'S COMPLEMENT	LTC
	ADD	A
	SUBTRACT	S
	COMPARE SIGNED	C
	MULTIPLY	M
	DIVIDE	DV
Logical	LOAD ONE'S COMPLEMENT	LOC
	AND	N
	OR	O
	EXCLUSIVE OR	XO
Bit Manipulation	SET BIT UPPER BYTE	SBU
	SET BIT LOWER BYTE	SBL
	CLEAR BIT UPPER BYTE	CBU
	CLEAR BIT LOWER BYTE	CBL
	TEST BIT UPPER BYTE	TBU
	TEST BIT LOWER BYTE	TBL
Shift	SHIFT SINGLE	SFTS
	SHIFT DOUBLE	SFTD

Table III (continued)

Instruction Type	Instruction	Mnemonic
Program & Interrupt Control	BRANCH ON CONDITION	BC
	EXCHANGE STATUS AND PC	XSPC
	RETURN FROM INTERRUPT	RINT
Input/Output	REGISTER INPUT	RIC
	REGISTER OUTPUT	ROC

Extended Short Format Instructions

The extended short format duplicates the most common standard format operations but in a short format. As discussed earlier, this format aims at minimizing the program memory and the execution time for high frequency instructions. Figure 6, on the next page, explains the details of various extended short formats. The instruction set is given in Table IV.

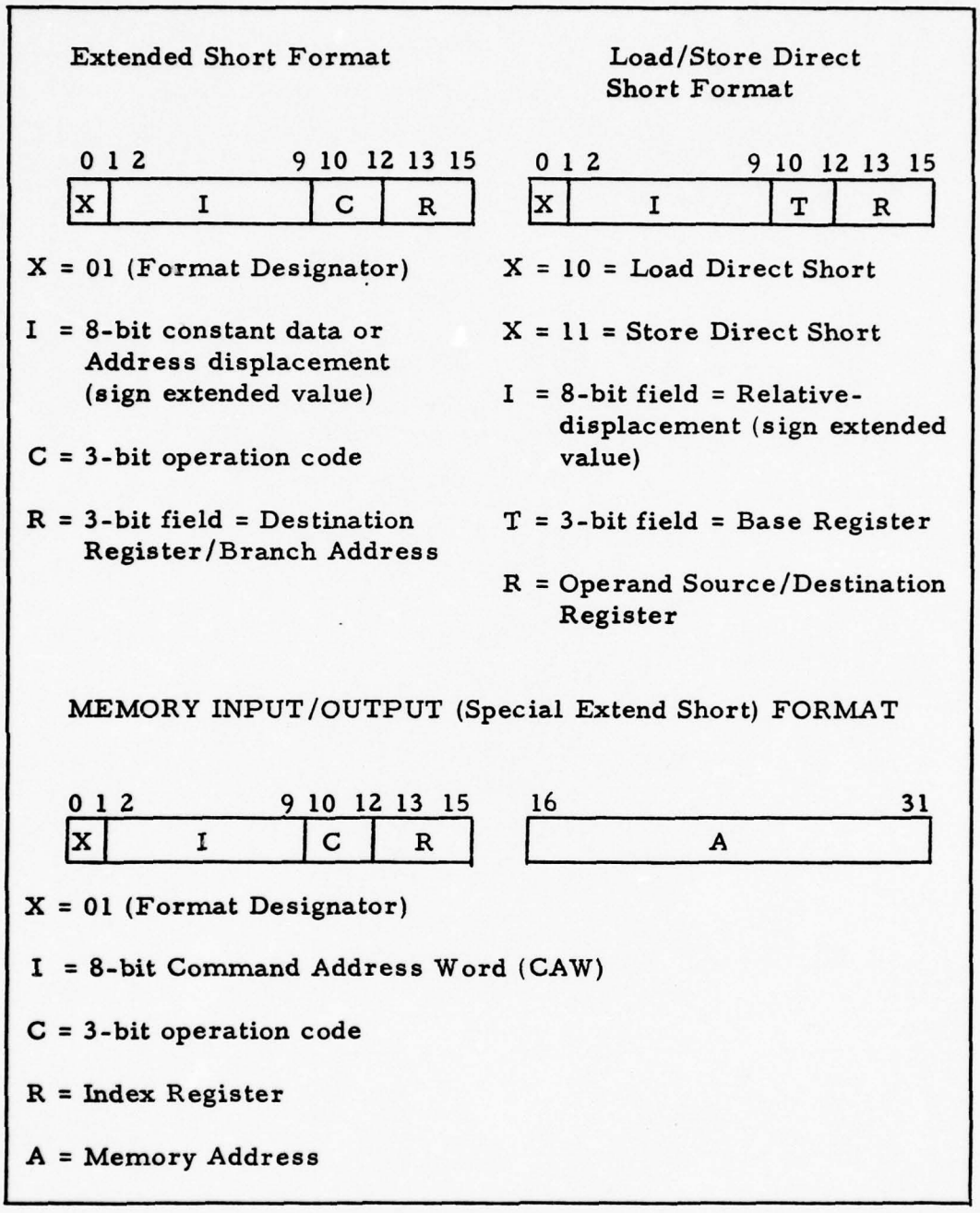


Fig. 6. Extended Short Formats

Table IV
Extended Short Format Instructions

Instructions	X-Field	C-Field	Mnemonic
LOAD DIRECT SHORT	10	--	LDS
STORE DIRECT SHORT	11	--	STDS
LOAD CONSTANT SHORT	01	000	LCS
ADD CONSTANT SHORT	01	001	ACS
COMPARE CONSTANT SHORT	01	010	CCS
BRANCH ON CONDITION-SHORT	01	011	BCS
BRANCH INDIRECT AND LINK REGISTER SHORT	01	100	BILR
INCREMENT AND BRANCH IF NEGATIVE SHORT	01	101	IBNS
MEMORY INPUT	01	110	MIC
MEMORY OUTPUT	01	111	MOC

Instruction Execution

Every DP/M instruction execution cycle can be viewed as occurring in two phases. The first phase is associated with obtaining an operand (from memory, a register or within the instruction), and the second phase is the use of that operand in conjunction with a second operand (a register or memory location) to effect the desired operation. This process of an instruction specifying two functions (i.e. selection of operand source and execution) has major advantages.

The software programmer has a basic set of instructions that can be used with a variety of source/destination operands. For example, an add instruction can have an implied destination register and multiple operand sources (another register, contents of a memory address that may be indexed, or an immediate data value imbedded within the instruction). This flexibility of specifying a wide variety of operand types not only adds power to the use of the instruction, it also simplifies the control logic required to implement a given set of instructions. Whenever an instruction is fetched from memory and ready for decode, a predefined set of fields within the instruction can be decoded to derive the necessary operands. The operand derivation can be independent of the instruction execution cycle and, hence, common to all operations. Once the operands have been derived and placed in the appropriate internal working registers, one common instruction execution cycle can be invoked to perform the operation using the internal working registers for operands.

Interrupt Control Structure

The interrupt structure of the DP/M PE is distributed between the processor, I/O, and BIU. Each portion controls that part of the interrupt structure that is appropriate to it. As such, the processor controls its own internal interrupts (overflow, invalid Op-code, I/O and memory time-out) and all interrupt masks. The internal and external interrupt masks are contained in the processor status word, as shown in Fig. 7.

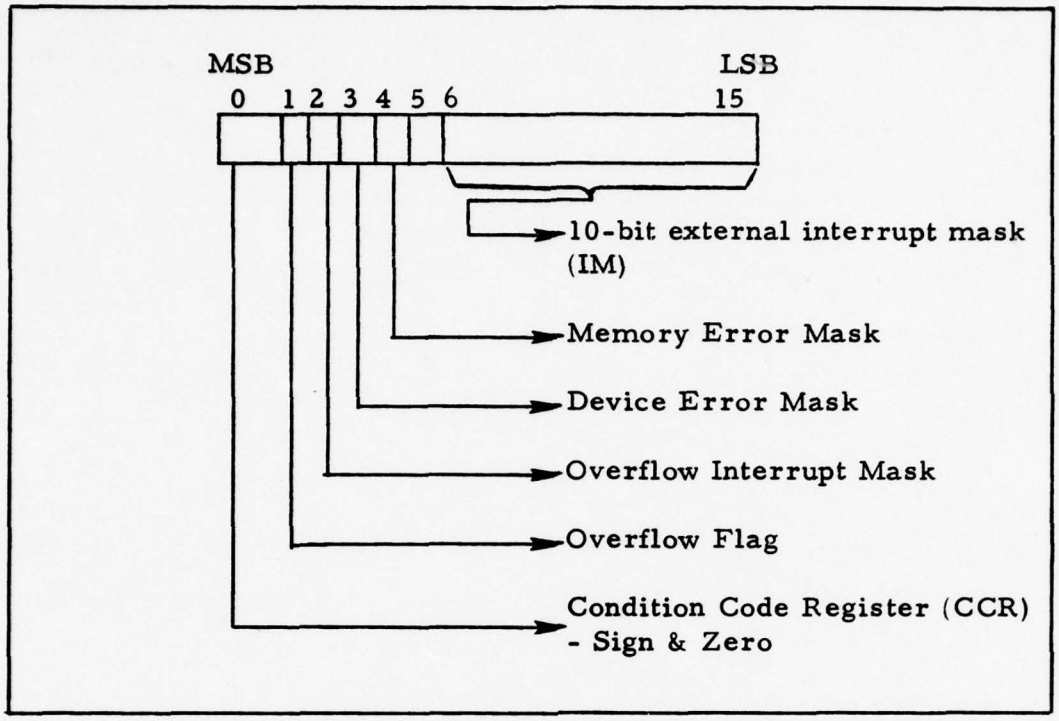


Fig. 7. Status Word Format

The Status Word (SW) contains a two-bit condition--code register, an overflow indicator and interrupt mask, a device, error interrupt mask, a memory error interrupt mask, and 10-bits of mask for interrupts external to the processor (i.e., BIU and I/O). These later 10-bits are external to the processor chip and reside physically on the I/O and BIU chips, and are accessed by the processor performing an I/O operation with a Command Address Word (CAW) of 00 (Hex) as a part of any status word read/write instruction. Thus, whenever a status word read or write instruction is executed, the processor directly accesses its own 6-bits, and accesses the remaining external 10 bits by simulating an I/O

instruction with a CAW of 00 (Hex), as if external interrupt mask (IM) were an I/O device.

Interrupt Context Switching

As with the interrupt mask, the interrupt initiated response is distributed between the processor, I/O and BIU. The processor supports internally and externally vectored interrupts. When an interrupt occurs, the current program counter and status word are saved in a memory stack using interrupt stack pointer (R_6). The new values of PC and SW are loaded from a pair of memory trap locations unique to the particular interrupt. In the case of internally generated interrupts, the processor determines the trap locations. For all external interrupts, the trap locations are specified by the interrupting unit in response to the "interrupt acknowledge."

I-Bus Interface Unit

As mentioned earlier, the processor communicates with memory, BIU, and I/O device through the I-Bus. The processor is a "Bus Master," i.e., it can take control of the I-Bus whenever it wants to transmit data to/receive data from a "slave" device like main memory. It, however, has the lowest priority for getting the bus control.

When required, the processor initiates a bus master assignment by issuing bus request (BRQ) and by specifying whether it is a send cycle (DRCV = 0) or a receive cycle (DRCV = 1). At the same

time the processor also specifies whether it wants to communicate with memory (IOSL = 1) or an I/O device (IOSL = 0). When initiating this assignment, the processor ensures that the 16-bit data and/or 16-bit address are available on the respective lines. If no other master of higher priority is in control of the I-Bus, the processor gets the control. This is indicated by "BUS GRANT" signal going HIGH at the processor end. In response to this, the processor generates 4-bit ID (0000), transfer request (TRQ), and the bus release (BREL). The bus release signal removes the bus request signal, thus allowing the masters to compete for the bus control. The new bus assignment, however, will wait until the processor removes its TRQ signal.

All slave devices connected to the bus, receive TRQ and decode the address to determine which slave is being addressed. If the processor indicates a send cycle, the device after decoding the address as valid, asserts transfer acknowledge (TACK) and clocks the data from the I-Bus into its register. In the case of memory, the TACK is delayed until the memory write cycle is complete. If the processor indicates a receive cycle, the device after decoding the address, starts sending data. In the case of the memory module, it would mean starting the read cycle. After putting data on the I-Bus, the memory will generate TACK.

The processor uses TACK to release the TRQ after delaying the TACK for the worst case of I-Bus skew (150 n.s.). In case of

a receive cycle, the processor clocks the data from the I-Bus into a data buffer.

If the processor attempts to address a nonexistent memory location, a watchdog timer, in memory control unit, generates "Transfer Time Out (TTO)" signal. The processor treats TTO like an ordinary TACK, but in addition, it sets the "Memory Error" flag.

In case of external interrupts, the external devices issue an "Interrupt Request (IRQ)." The processor acknowledges the external interrupt by issuing the "Interrupt Acknowledge (IAK)." On receiving IAK, the highest priority device places the address of its interrupt trap-vector location on the data lines and issues "Transfer Acknowledge (TACK)." The processor loads the address (trap-vector) and removes the interrupt acknowledge signal. It, then, proceeds to service the interrupt request.

Summary

In this chapter, a functional description of the processor requirements has been presented. Various instruction formats and addressing modes are outlined and the processor instruction set is tabulated. A discussion on interrupt scheme and I-Bus is also included.

The next chapter discusses the salient features of Am 2900 microprocessor family and the implementation of the processor using that hardware.

III. Processor Hardware and Implementation

This chapter discusses the Am 2900 microprocessor family. The Central Processor Unit (CPU), Microprogram Control Unit, and the Interrupt Control Unit chips are described in detail, highlighting the features which are needed to meet the emulation requirements of the processor. The remainder of the chapter is concerned with the processor implementation.

The Am 2900 Family

The Am 2900 is a 4-bit, bipolar, bit-slice microprocessor family utilizing low-power Schottky TTL technology. The basic family developed by Advanced Micro Devices consists of the 4-bit CPU slice (Am 2901), the Microprogram Sequencer (Am 2909), and the Next-address Control Unit (Am 29811). The functions of Am 2909 and Am 29811 have been combined into a newer chip (Am 2910), called the Microprogram Control Unit (Ref 3:1). Many direct-support circuits, like Am 2902 Look-ahead Carry Generator, and Am 2914 Vectored-interrupt Controller, can be added to a specific design repertoire. The four-bit slice architecture provides a good compromise between expandability and the package-count. It is estimated that to build a micro-computer based on Am 2900, will require 30 or more packages (Ref 2:2). Virtually any machine can be implemented using 2900 microprocessors.

Am 2901 Central Processing Slice

As mentioned earlier, Am 2901 is a four-bit CPU slice cascadable to any number of bits. Its two major elements are the 16-word by 4-bit, 2-port Random Access Memory (RAM), and a high speed Arithmetic and Logic Unit (ALU). Figure 8, on the next page, illustrates the detailed architecture of this chip.

Data from any of the 16-words in RAM can be read from the A-port or B-port of the RAM by controlling the 4-bit A-address field or the B-address field. New data, however, is always written into the word defined by the B-address field. The RAM data input field is driven by 3-input multiplexers. This configuration allows the ALU output data to be shifted left or right by one bit position or not shifted in either direction. The RAM output data is held by two latches to avoid any race condition when new data is being written.

The ALU receives input data from RAM A-port, B-port, D inputs, and from the Q-register. The D input is a 4-bit wide direct data field which can be used to insert data into the working registers or to modify any of the internal data files. The Q-Register is a separate 4-bit file intended primarily for multiplication and division routines but can also be used as a general purpose accumulator. The ALU itself is a high-speed arithmetic/logic operator capable of performing three binary arithmetic and five logic functions (Ref 2:8). The ALU data output can be routed to several destinations. It can be a data output of the device and/or it can also be stored in the RAM or

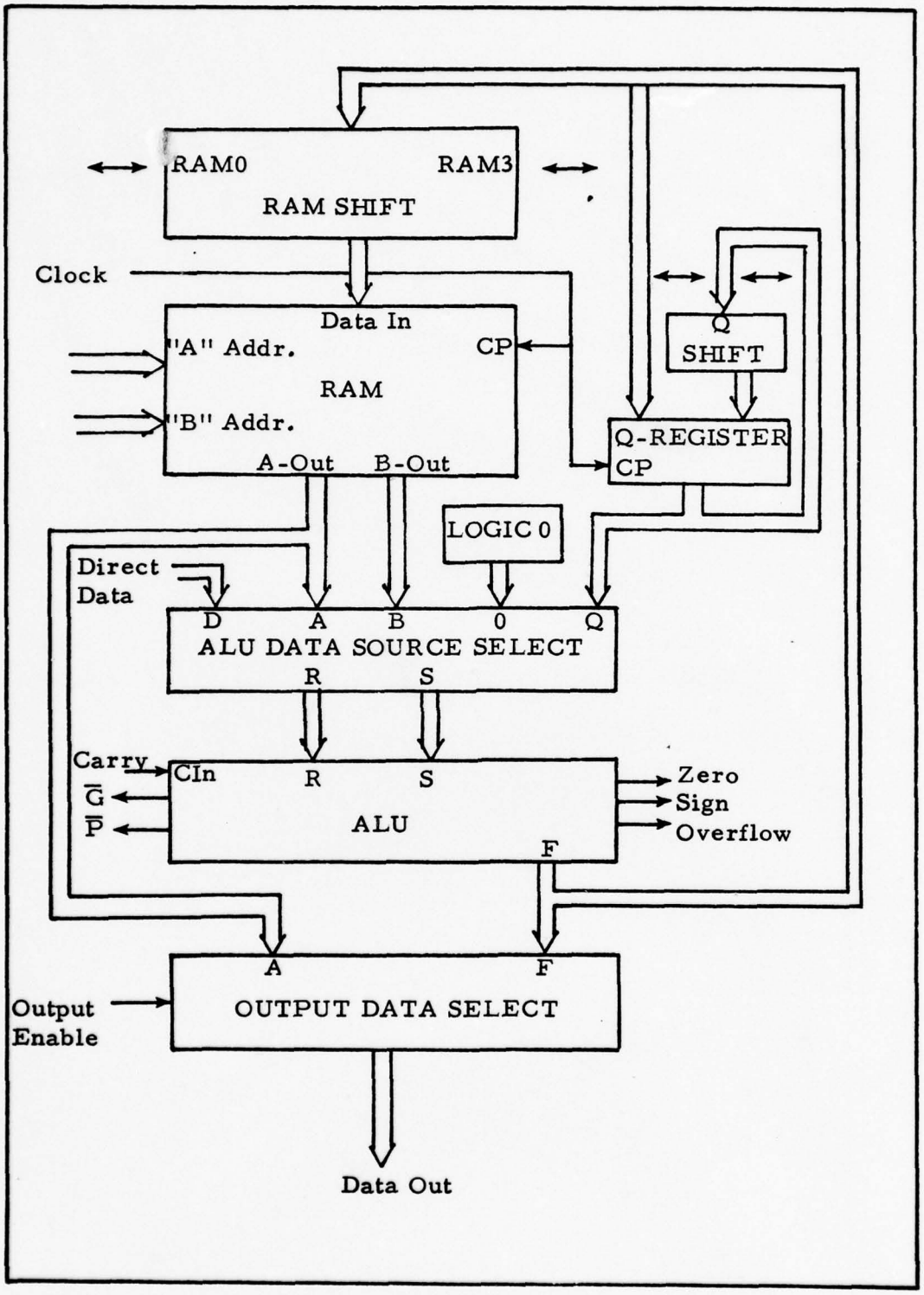


Fig. 8. Architecture of Am 2901

the Q-register. There are eight combinations of ALU data sources, eight ALU functions, and eight combinations of destinations for the ALU output data. Three 3-bit microfields are used to control source, function and destination parameters (Ref 2:8, 9).

The ALU has three other status-oriented outputs. These are F_3, F = 0, and overflow (OVR). The F_3 output is the most significant (sign) bit of the ALU. The F = 0 output is used for zero detect. The overflow output (OVR) is used to flag arithmetic operations that exceed the available two's complement number range. The normal technique for cascading the ALU of several devices is in a look-ahead carry mode. Carry generate, $\overline{G}$ and carry propagate, $\overline{P}$, are outputs of the device for use with a carry-look-ahead generator like Am 2902 (to be described later in this chapter).

Am 2910 Microprogram Controller

The Am 2910 microprogram controller is an address sequencer intended for controlling the sequence of execution of microinstructions stored in microprogram memory. Besides the sequential access capability, it provides conditional branching to any microinstruction within its 4-K micro-word range. A 5-level last-in, first-out-stack provides microsubroutine return linkage and looping capability. A microinstruction loop counter is also provided with a count capacity of 4096. Figure 9 shows the architecture of this chip.

During each microinstruction, the microprogram controller provides a 12-bit address from either a microprogram counter

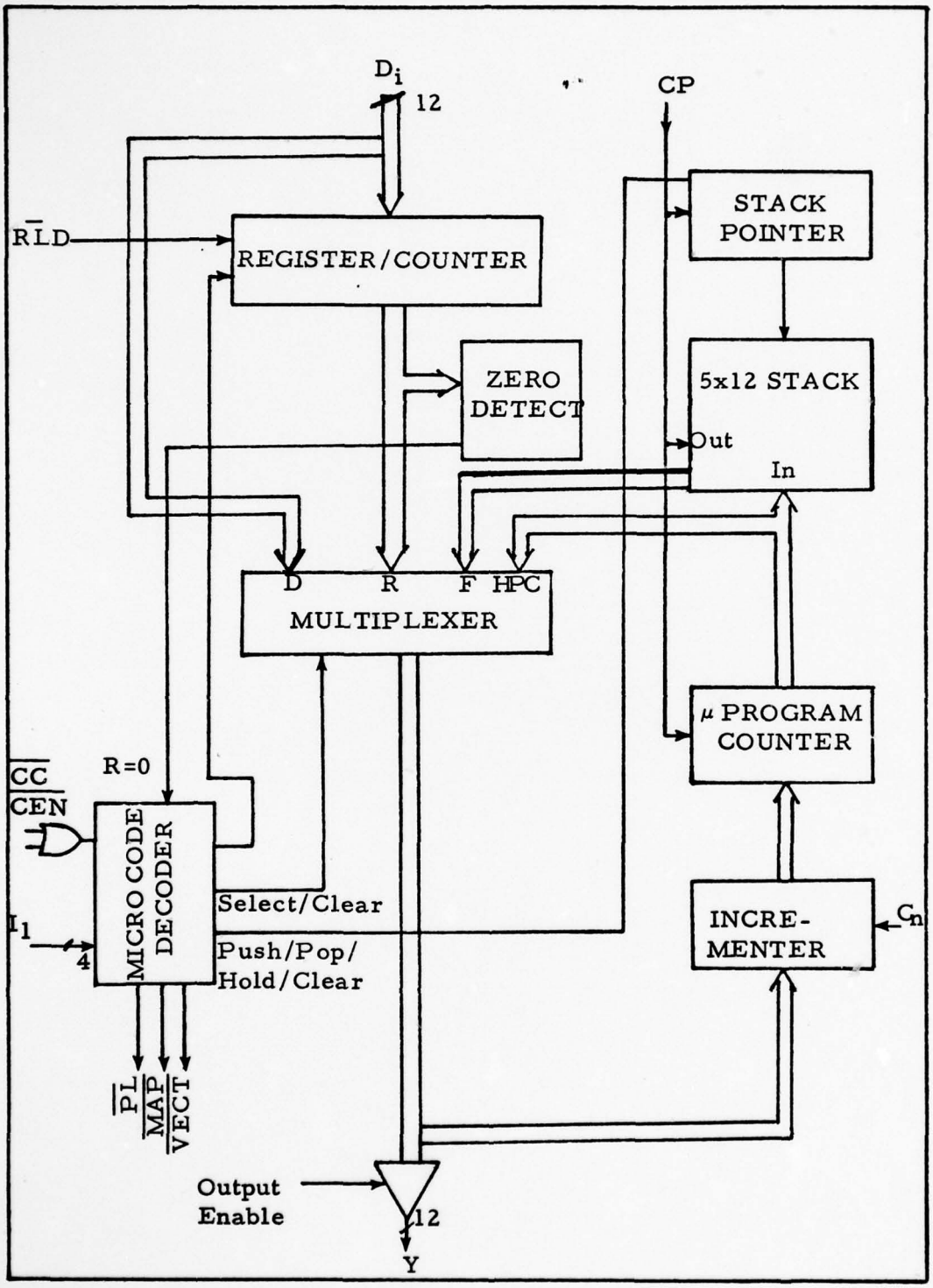


Fig. 9. Architecture of Am 2910

register, or an external (direct) input, or a register/counter (which retains data loaded during a previous microinstruction), or a five-deep stack.

The register/counter loads direct data (on a positive clock transition) when its load control, $\overline{\text{RLD}}$, is LOW. The output of the register/counter is available (if selected) as a source for the next microinstruction address.

The microprogram counter is composed of a 12-bit incrementer followed by a 12-bit register. When the "carry-in" to the incrementer is HIGH, the microprogram register is loaded on the next clock cycle with the current Y output word plus one. Sequential microinstructions are thus executed. When the "carry-in" is LOW, the incrementer passes the Y output word unmodified so that the previous value is reloaded. The same microinstruction is thus executed any number of times.

The stack is used to provide return address linkage when executing microsubroutines or loops. The stack contains a built-in stack pointer which always points to the last file word written.

A four-bit microfield is used to control the 16 microinstructions which describe the various functions of the microprogram controller (Ref 3:3). For each of these instructions, one of the three outputs $\overline{\text{PL}}$, $\overline{\text{MAP}}$, and $\overline{\text{VECT}}$ is LOW. These signals may be used to enable 12-bit data either from the pipeline register, from a mapping PROM, or from a third source to be fed at the direct (D) inputs of

the controller.

Am 2902 Look-ahead Carry Generator

The Am 2902 is a high-speed, look-ahead carry generator which accepts up to four pairs of carry propagate ($\overline{P}$) and carry generate ($\overline{G}$) signals and a carry input. It provides anticipated carries across four groups of binary ALUs. The device also has carry propagate and carry generate outputs which may be used for further levels of look-ahead. The Am 2902 is generally used with the 2901 microprocessor units to provide look-ahead over more than four bits. Figure 10 shows four 2901s connected to an Am 2902.

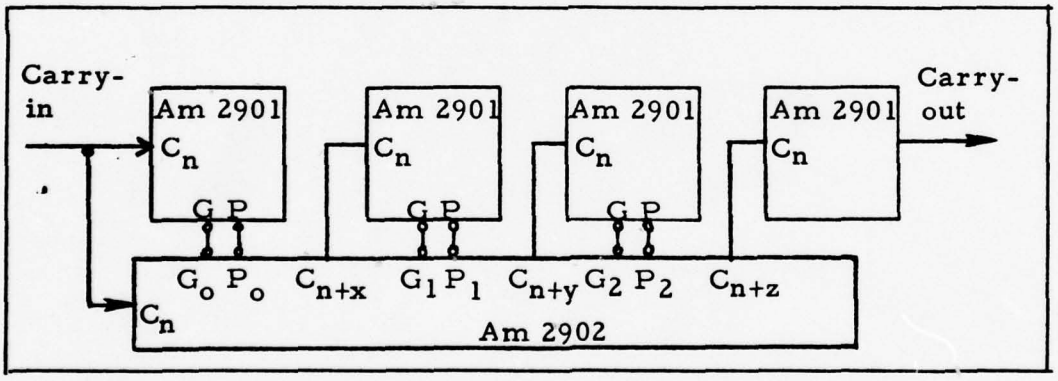


Fig. 10. 16-bit Carry Look-ahead Connection (Ref 2:25)

Am 2914 Vectored Priority Interrupt Controller

The Am 2914 is a high speed, 8-bit, cascadable priority unit. It is recommended to be used in conjunction with Am 2900 family microcomputer designs (Ref 4:2). Figure 11 shows the functional block diagram indicating various inputs and outputs of this chip.

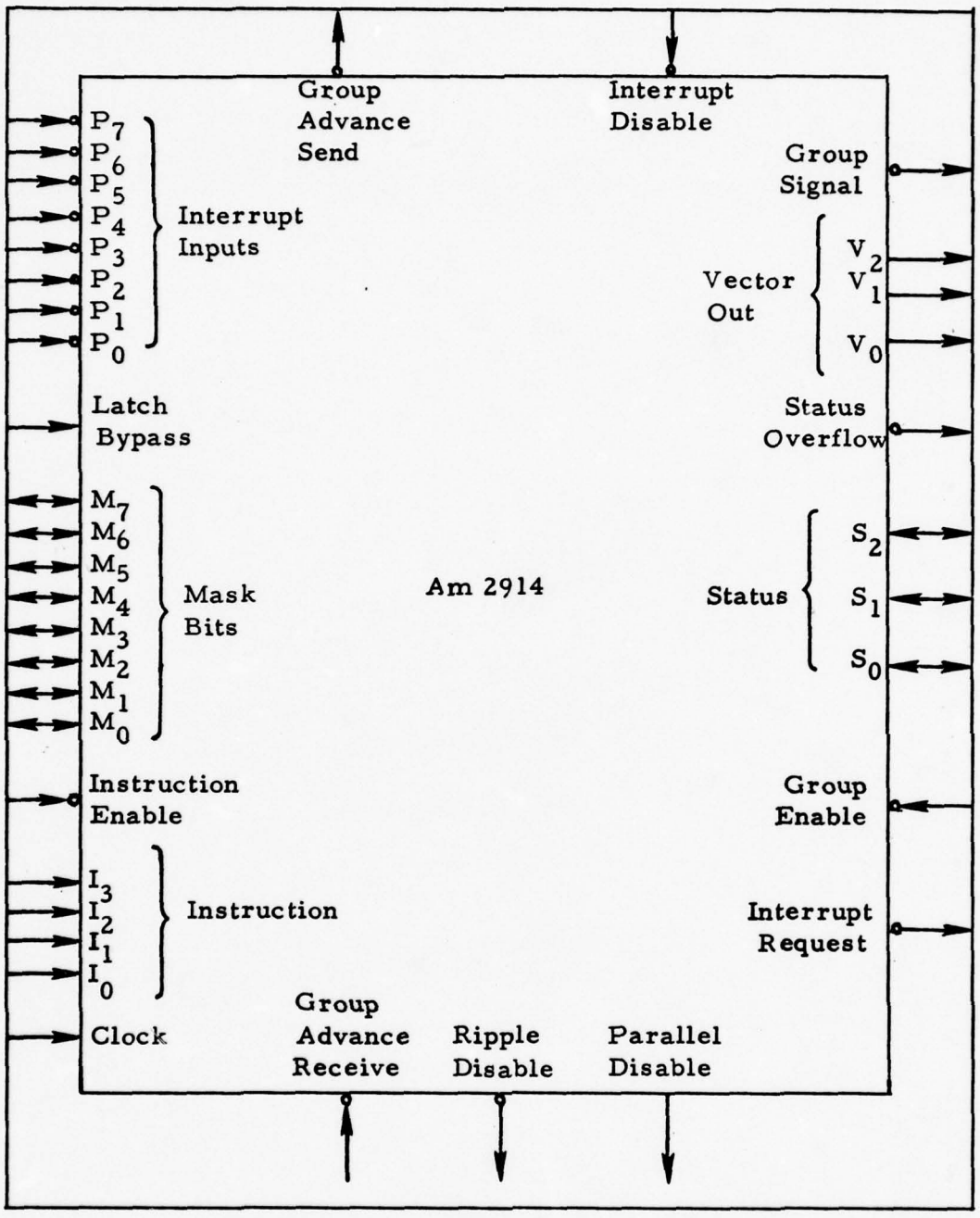


Fig. 11. Vectored Priority Interrupt Controller (Ref 4:2)

The Am 2914 receives interrupt requests on 8 interrupt input lines (P_0-P_7). An internal latch may be used to catch pulses on

these lines, or the latch may be bypassed so the request lines drive the edge-triggered interrupt-register directly. An 8-bit mask register is used to mask individual interrupts. Interrupt inputs are internally ANDed with the corresponding bits in the mask register and the results are sent to an 8-input priority encoder, which produces a 3-bit encoded vector representing the highest numbered input which is not masked. An internal status register is used to point to the lowest priority at which an interrupt will be accepted. The contents of the status register are compared with the output of the priority encoder, and an interrupt request output will occur if the vector is greater than or equal to the status. Whenever vector is read from the Am 2914, the status register is automatically updated to point to one level higher than the vector read. Signals are provided for moving the status upward across devices (Group Advance Send and Group Advance Receive) and for inhibiting lower priorities from higher order devices (Ripple Disable, Parallel Disable, and Interrupt Disable).

The Am 2914 is controlled by a 4-bit microfield (Ref 4:13). The command on the instruction lines is executed if Instruction Enable Control is LOW and ignored otherwise.

Processor Implementation

The processor was implemented using the Am 2900 family hardware. Figure 12 shows the functional block diagram of the

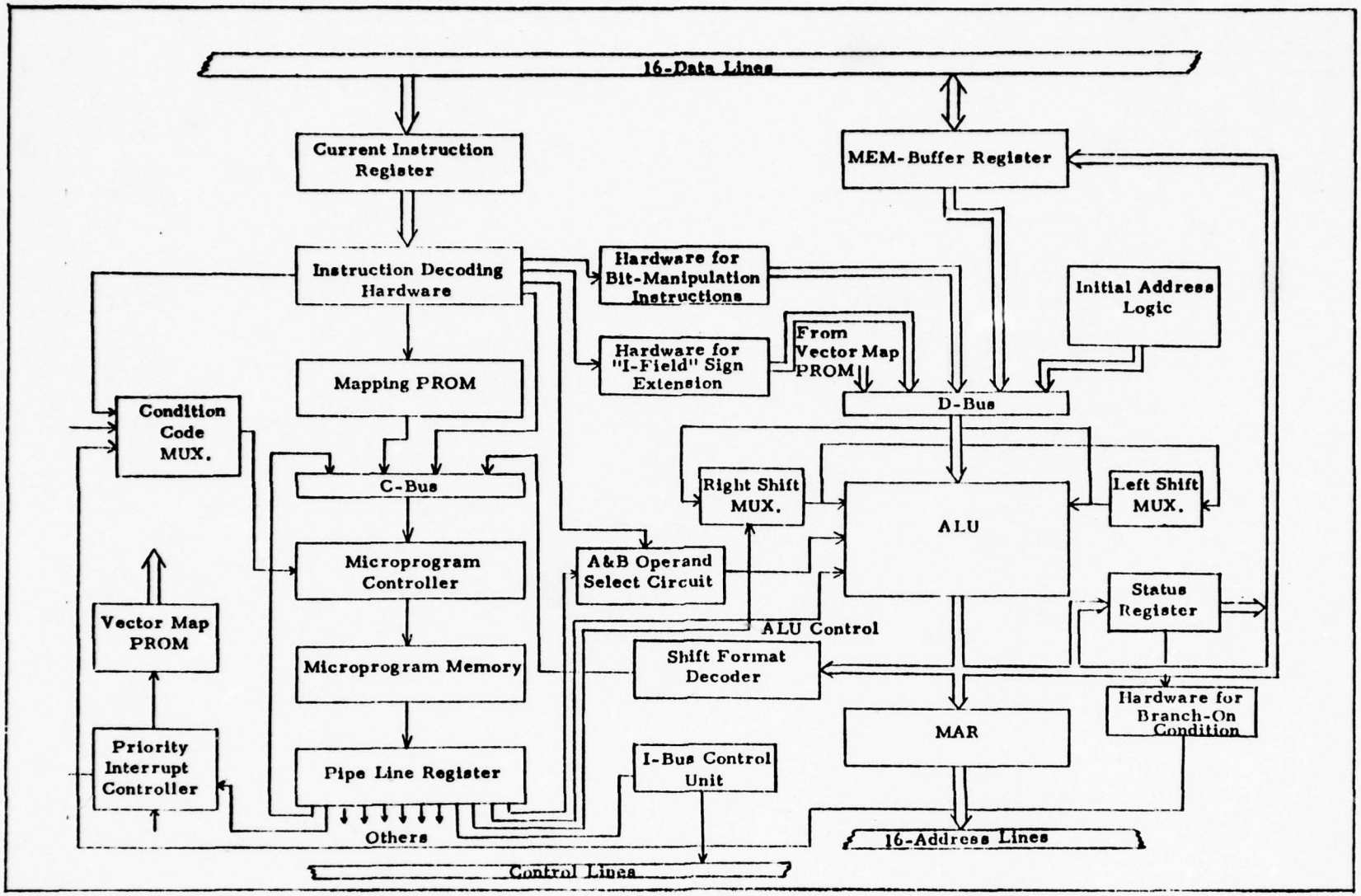


Fig. 12. Functional Diagram of the Processor

processor. A blockwise detailed description is presented in the subsequent paragraphs.

Current Instruction Register (CIR) and Instruction Decoding. The details of the CIR and instruction decoding scheme are shown in Fig. 13. The CIR loads data under the control of a signal from the I-Bus control unit (to be described later in this chapter). It uses Am 2918, a quad D-Register with Standard (TTL) and three-state outputs, as the basic element. The standard outputs are applied to two decoders and one PROM for decoding the various fields. The decoders and PROM are permanently enabled. This causes the results of decoding to be available continuously as long as an instruction is held in CIR. The three-state outputs of Am 2918 are enabled whenever the contents of CIR are to be monitored.

Decoding the "X" and the "M" fields, decides on one of the seven addressing modes (described in Chapter II) to be followed for calculating the Derived Address or the Derived Operand. The Operation Codes for the Standard format instruction set need to be reformatted so that some attribute (inherent in the OP-Code) can be used to decide whether Derived Address is to be calculated or the Derived Operand is to be computed for accomplishing the function indicated by an OP-Code. PROM A reformats the OP-Code C1. Corresponding to each OP-Code needing the calculation of Derived Address, the PROM A outputs an 8-bit field with One as the MSB, otherwise MSB is zero. Thus, the MSB of the reformatted OP-Code is applied to the

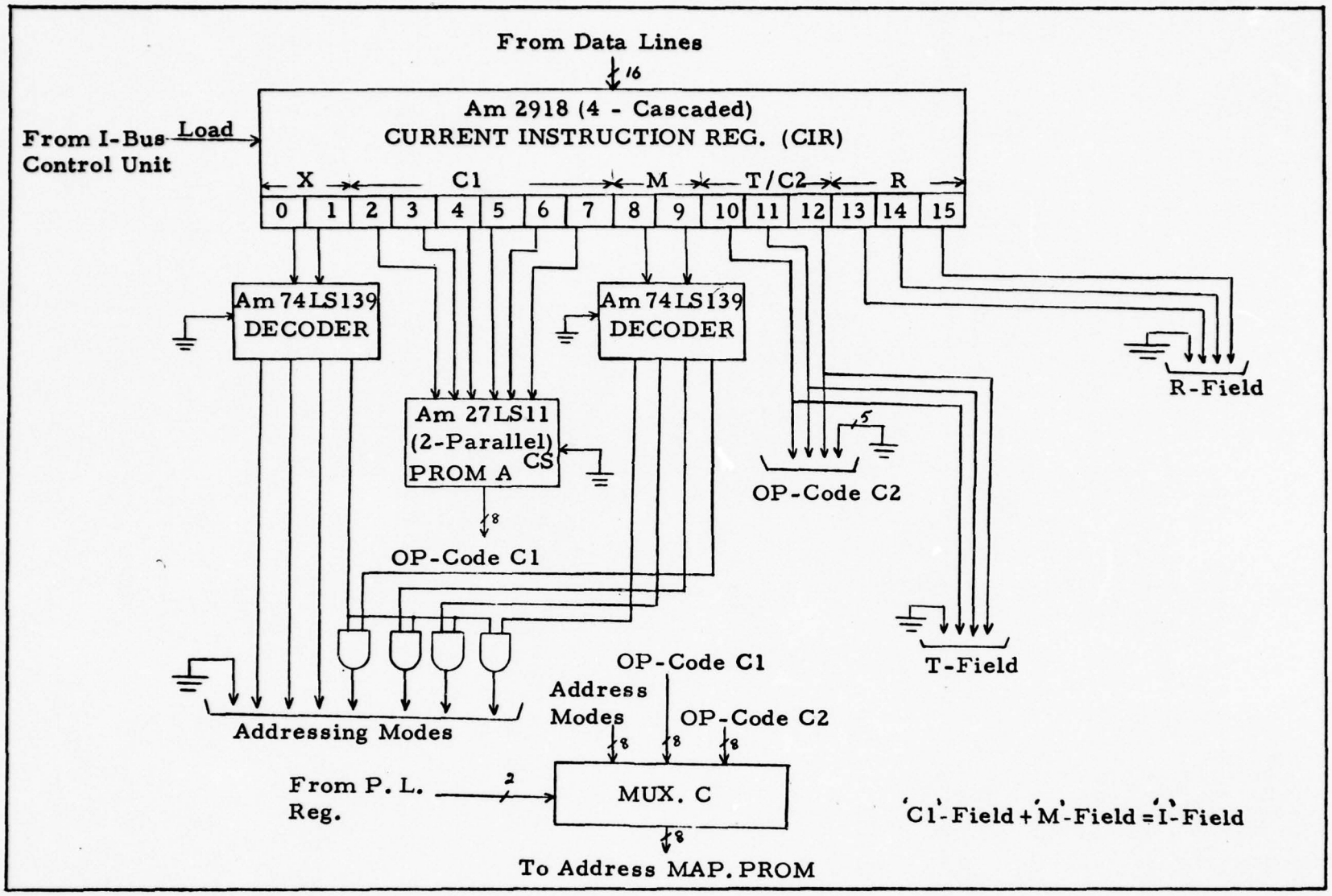


Fig. 13. Current Instruction Register

Condition-Select Multiplexer for making the appropriate decision.

The specified OP-Codes, along with their reformatted values, for the Standard Format Instruction Set have been listed in Table V.

Table V
Standard Format OP-Code Reformatting

Instruction	Specified OP-Code	Reformatted OP-Code
LOAD	000001	00000001
MOVE & AUTOINCREMENT	010010	00010010
PUSH	010001	00010001
LOAD 2'S COMPLEMENT	010011	00010011
ADD	000011	00000011
SUBTRACT	000100	00000100
COMPARE SIGNED	000101	00000101
MULTIPLY	000110	00000110
DIVIDE	000111	00000111
LOAD 1'S COMPLEMENT	001000	00001000
AND	001001	00001001
OR	001010	00001010
EXCLUSIVE OR	001011	00001011
SHIFT SINGLE	001100	00001100
SHIFT DOUBLE	001101	00001101
EXCHANGE STATUS AND PROGRAM COUNTER	001111	00001111

Table V (continued)

Instruction	Specified OP-Code	Reformatted OP-Code
REGISTER INPUT	100011	00100011
REGISTER OUTPUT	100100	00100100
STORE	000010	10000010
STORE THROUGH MASK	010000	10010000
PUSH MULTIPLE	110001	10110001
POP MULTIPLE	110010	10110010
SET BIT UPPER BYTE	010111	10010111
SET BIT LOWER BYTE	011000	10011000
CLEAR BIT UPPER BYTE	011001	10011001
CLEAR BIT LOWER BYTE	011010	10011010
TEST BIT UPPER BYTE	011011	10011011
TEST BIT LOWER BYTE	011100	10011100
BRANCH ON CONDITION	001110	10001110
RETURN FROM INTERRUPT	100010	10100010

T-Field/C2-Field. As shown in Fig. 13, bits 10-12 in CIR represent either the T-field (for standard Format Instructions) or the C2-field (for extended short Format Instructions). In order to make OP-Code C2 an 8-bit field, 5-logic zero lines are included in the right-most position.

Table VI shows the specified as well as the 8-bit values of the OP-Code C2.

Table VI
Operation Codes for Extended Short Format

Instruction	Specified OP-Code	8-Bit OP-Code
LOAD CONSTANT SHORT	000	00000000
ADD CONSTANT SHORT	001	00100000
COMPARE CONSTANT SHORT	010	01000000
BRANCH ON CONDITION SHORT	011	01100000
BRANCH INDIRECT & LINK REGISTER SHORT	100	10000000
INCREMENT & BRANCH IF NEGATIVE SHORT	101	10100000
MEMORY INPUT	110	11000000
MEMORY OUTPUT	111	11100000

Sign Extension of the I-Field. In the case of Extended Short Format Instructions, bits 2-9 in CIR represent the I-Field. Before using, the I-Field needs to be right justified and sign-extended to make it compatible with the 16-bit data format. Since the use of these Short Format Instructions is expected to be quite extensive (they cut down the memory requirement by 25 to 30%), it is presumed that I-Field is always present and requires decoding. Figure 14 shows the circuit for this purpose. Bits 2-9 from the CIR are

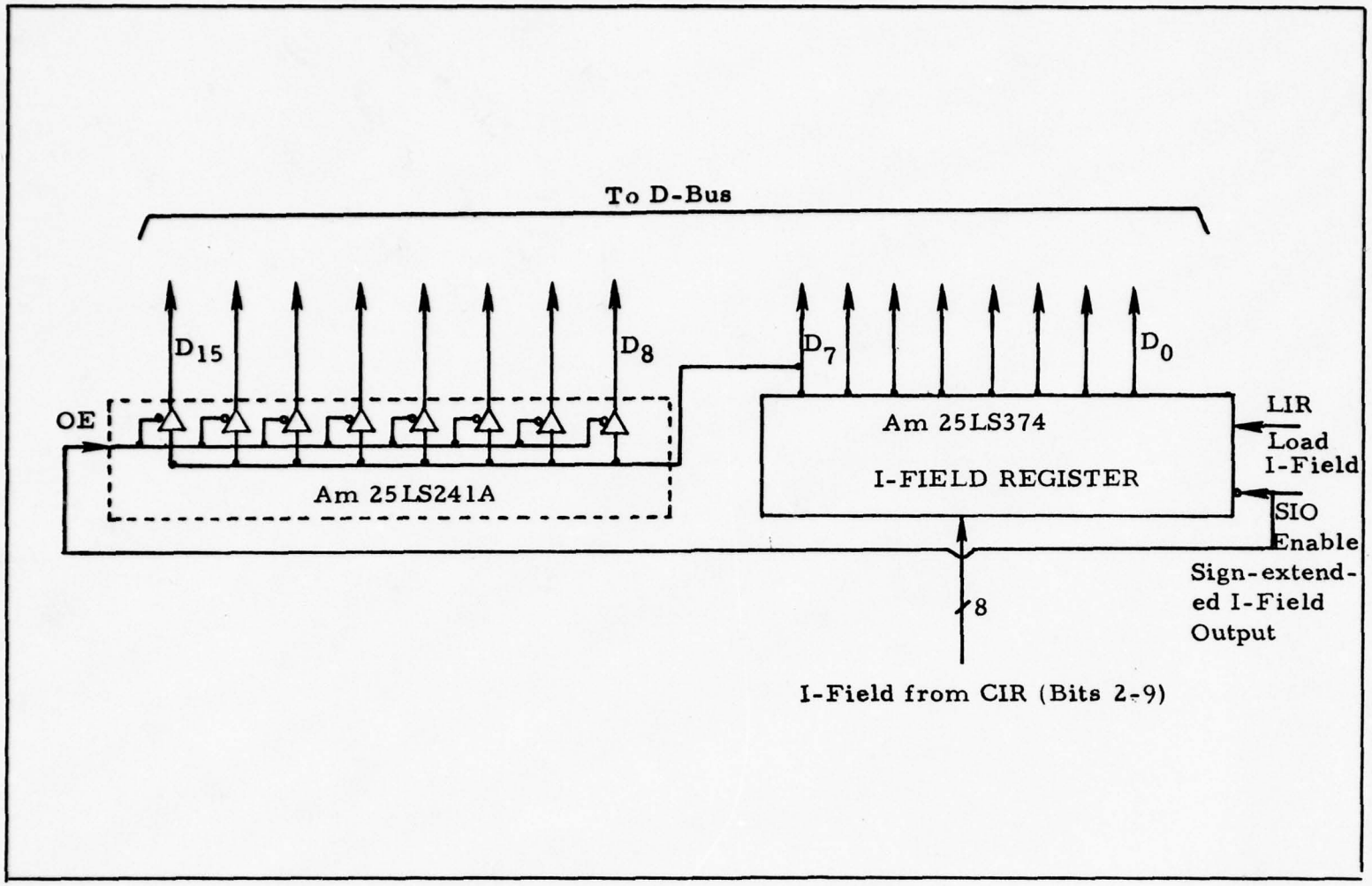


Fig. 14. Sign Extension of I-Field

clocked into the I-field register (Am 25LS374). The MSB from this register is applied to eight tristate buffers (Am 25LS241A) for sign extension. A common output enable control puts the sign-extended and right justified I-field on to the D-bus.

Hardware for Bit Manipulation Instructions. Bit manipulation instructions are a group of six instructions which specify a particular bit (either in lower byte or in upper byte) in a register to be tested, set or cleared. For these instructions the positional information about the bit (to be tested, set or cleared), is contained in the R-field. For instance if contents of R-field are 03 (Hex), then the 3rd bit of the lower byte or of the upper byte in the contents of a specified register is to be manipulated. For each bit position, therefore, a known 16-bit mask is needed which should have "1" in the bit position specified by the R-field and the rest all zeroes.

The mask can be ANDed/ORed with the contents of the given register to achieve the desired results. The LSB of OP-Code C1 specifies lower byte (LSB = 0) or the upper byte (LSB = 1). PROM B, as shown in Fig. 15, accomplishes the desired function.

Multiplexer-C. The Multiplexer-C, in Fig. 13, controls the application of the results of instruction-decoding to the subsequent circuitry, i.e. Address Mapping PROM. First of all, an 8-bit "Addressing Mode" field is applied to this PROM to start the calculation of either the Derived Address or the Derived Operand. Each of the seven "Addressing Modes" points to a unique starting address.

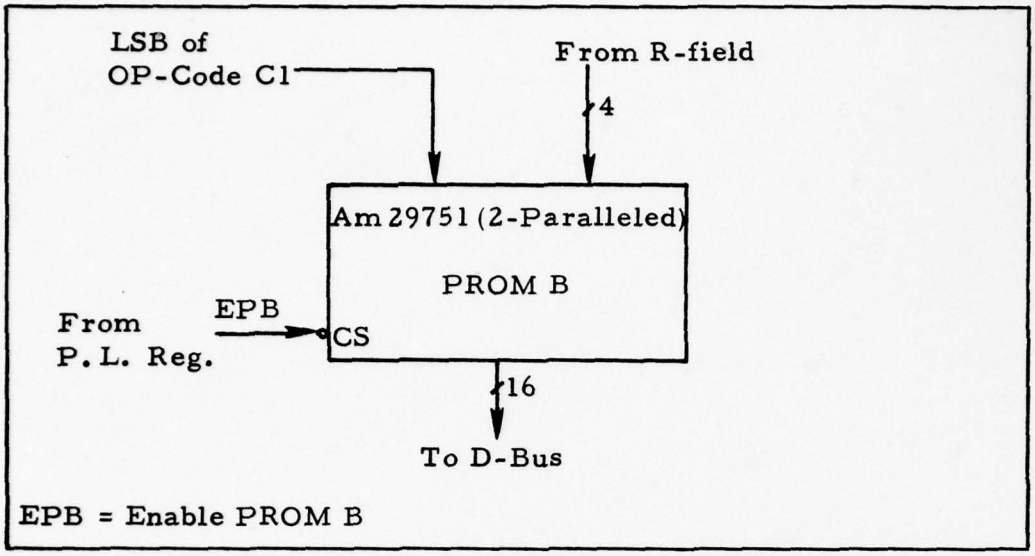


Fig. 15. Mask Generation for Bit Manipulation

After calculation of the Derived Address/Derived Operand, either OP-Code C1 or OP-Code C2 is used to point to a unique address for starting the execution phase. The OP-Code C1 is used for Standard Format Instructions, and C2 being used for Extended Short Format Instructions. A 2-bit field from the Pipeline Register controls the function of the Multiplexer-C as given in Table VII.

Table VII
Control Field for Multiplexer C

Micro Code		Mnemonic	Explanation
I_1	I_0		
0	0	x x	x x x
0	1	AM	Select Addressing Mode
1	0	OC1	Select OP-Code C1
1	1	OC2	Select OP-Code C2

Microinstruction Control Circuit

The microinstruction control circuit consists of an Address Mapping PROM, a Microprogram Controller, Microprogram Memory, and the Pipeline Register. Figure 16 shows the layout of these essential elements.

Address Mapping PROM. The Address Mapping PROM uses three Am 29761 chips connected in parallel. Each Am 29761 is a 256 words by 4 bits PROM. The 12-bit wide output from the Mapping PROM is needed to meet the input requirements of the Microprogram Controller (which is a 12-bit slice). This scheme gives the flexibility of addressing up to 4-K words in the Microprogram Memory, allowing the addition of more instructions at a later stage.

C-Bus. The C-Bus consists of four 12-bit fields; one each from Mapping PROM, Pipeline Register, Shift Format Decoding Circuit, and the T-field from CIR. Since all input lines are using Tristate Buffers, the 4-fields have been tied together and at any time only one of them is enabled to be applied to the D-inputs of the Microprogram Controller. C-Bus, thus, effectively performs like a 4-input multiplexer.

Microprogram Controller. The internal architecture and working of the Microprogram Controller (Am 2910) was explained earlier in this chapter. Basically, it controls the sequential flow of the microinstructions which may be initiated by any of the four starting address fields applied to the C-Bus. The conditional branch function

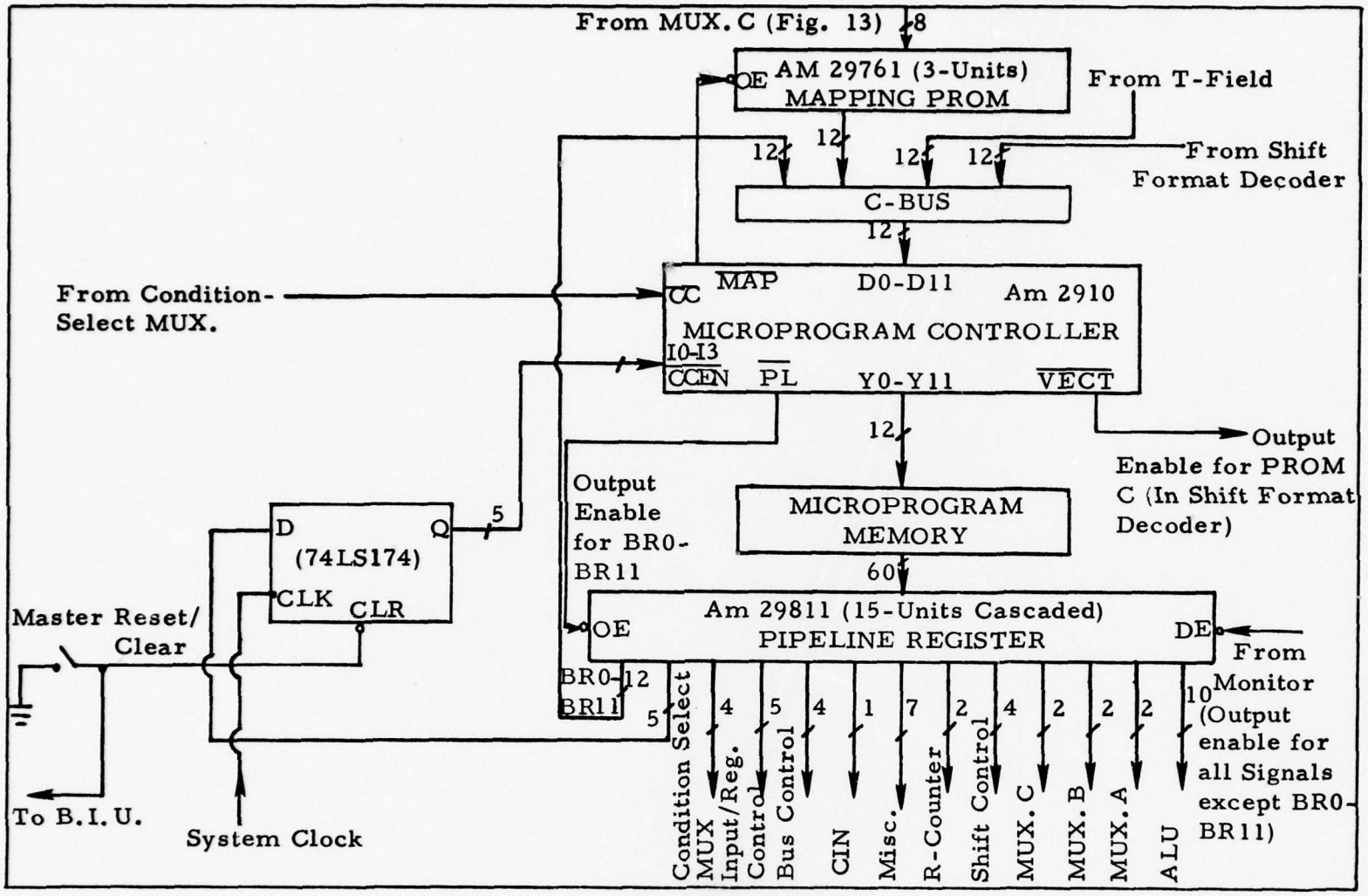


Fig. 16. Microinstruction Control Circuit

is accomplished in conjunction with a condition-select multiplexer. This multiplexer checks for a given condition and supplies the result at $\overline{CC}$ input of the Microprogram Controller. If the condition is true (logic 0 at $\overline{CC}$), a branch is made to a given microaddress (anywhere in the 4-K micro memory), otherwise the Microprogram Controller executes the next microinstruction in sequence. $\overline{CCEN}$ provides an over-riding control for $\overline{CC}$ input. If $\overline{CCEN}$ is made HIGH, $\overline{CC}$ is ignored and the subsequent action takes place as though $\overline{CC}$ were true.

A 4-bit field from the Pipeline Regiser (P. L. Reg) provides 16 microinstructions governing the various functions of the Microprogram Controller. With each microinstruction, the Program Controller provides an enable signal either to 12-MSB in the PL Reg or the Mapping PROM or a third source called vector. In this design the third source is PROM C used in Shift Format Decoding Circuit. Table VIII lists the microinstructions applicable to the Program Controller.

Condition-Select Multiplexer. The Condition-Select Multiplexer (SN 74150) is a 16-line to 1-line selector. Figure 17 shows the various conditions applied at its inputs. A brief description of each condition is given in the following paragraph.

The Internal Interrupt is generated by the interrupt control unit (Am 2914). The External Interrupt is generated by the external devices. The "Address"/"Operand" (AO) input is the result of

Table VIII
Control Instructions for Microprogram Controller
(Next Address Control Field)

Micro Code				Mnemonic	Instruction	Enable
I_3	I_2	I_1	I_0			
0	0	0	0	JZ	Jump to Address Zero.	PL
0	0	0	1	CSP	Cond. jump to subroutine; address in P. L. Reg.	PL
0	0	1	0	JMA	Jump to address at MAP. PROM output.	MAP
0	0	1	1	CJP	Cond. jump to address in P. L. Reg.	PL
0	1	0	0	PSH	Push stack and conditionally load counter.	PL
0	1	0	1	SRP	Cond. jump to subroutine; ADDRESS IN "R"/P. L. Reg.	PL
0	1	1	0	CJV	Cond. jump to vector address.	VEC
0	1	1	1	JRP	Cond. jump to address in "R"/P. L. Reg.	PL
1	0	0	0	RFC	Repeat loop if counter ≠ 0.	PL
1	0	0	1	RPC	Repeat P. L. Reg. address if counter ≠ 0.	PL
1	0	1	0	RTN	Cond. return from subroutine.	PL
1	0	1	1	JPP	Cond. jump to P. L. Address and pop stack.	PL
1	1	0	0	LCC	Load counter and continue.	PL
1	1	0	1	LP	Test end of loop.	PL
1	1	1	0	CON	Continue.	PL
1	1	1	1	TWB	Three-way branch.	PL

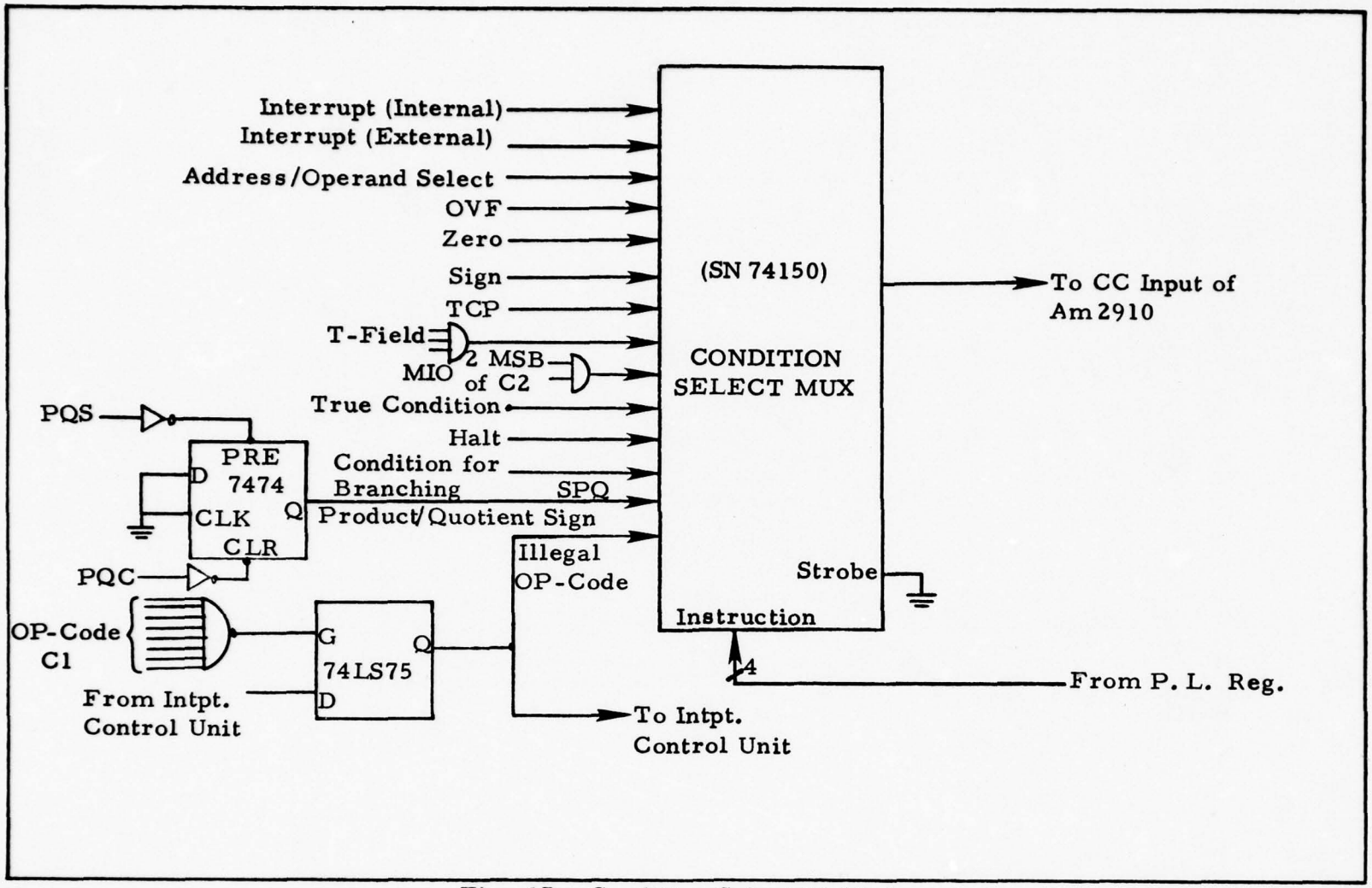


Fig. 17. Condition Select Multiplexer

checking the MSB of OP-Code C1 (formatted value) to decide the calculation of Derived Address/Derived Operand . Overflow, zero and sign inputs come from the status word register.TCP (Transfer Complete) is a signal from the I-Bus Control Unit. It indicates the completion of transfer of data between the Processor and a slave device. If in an instruction the M-field is 10, then the value of T-field is checked to decide the addressing mode. If T = 7, then the next word is to be brought from the memory to provide 16-bit data. If $T \neq 7$, then it indicates "Register Indirect Autoincrement" addressing mode. The next conditional input consists of 2 MSB of OP-Code C2. If this input is HIGH, it indicates a Memory input/output instruction. HALT input, if true, creates one instruction loop and indicates a waiting state. The result of checking the various conditions for branching is also provided as an input to this multiplexer. An illegal OP-Code is detected if all the 8-bits of C1 are "1." This condition is checked before execution phase for every instruction. A "true condition" is also used. This gives the flexibility of using the conditional jumps (Ref Table VIII) for Microprogram Controller as unconditional jump instructions. The output of the Condition-Select Multiplexer is applied to the $\overline{CC}$ input of the Am 2910 chip.

Table IX lists the micro codes for the selection of various conditions.

Table IX
Condition Selection Control Field

Micro Code I_3	I_2	I_1	I_0	Mnemonic	Condition Selected
0	0	0	0	INI	Internal Interrupt
0	0	0	1	INE	External Interrupt
0	0	1	0	AO	Derived Address/Derived Operand
0	0	1	1	OVF	Overflow
0	1	0	0	ZERO	Zero
0	1	0	1	SIN	Sign
0	1	1	0	TCP	Transfer Complete
0	1	1	1	T = 7	T = 7
1	0	0	0	MIO	Memory Input/Output
1	0	0	1	TC	True Condition
1	0	1	0	HLT	Halt
1	0	1	1	BOC	Branch-on One of 8-Conditions
1	1	0	0	IOC	Illegal OP-Code
1	1	0	1	SPQ	Output of Product/Quotient Flip Flop
1	1	1	0	x	x
1	1	1	1	x	x

Pipe-Line Register. As shown in Fig. 16, the Pipe-Line Register receives a 64-bit format from the Microprogram Memory and passes them as control signals to the various units in the Processor. It consists of 15-Am 2918 chips cascaded. The

"Enable" signal for the 12-MSB (BR0-BR11) is provided by the Microprogram Controller. The "Enable" signal for the rest of the P. L. Register comes from the "Monitor Control" Unit (to be discussed in Chapter V).

The description of each control field has been included in the discussion of the corresponding functional unit. The microinstruction format, as a whole, will be discussed in the next chapter.

Memory Buffer Register (MBR) and D-Bus

The Memory Buffer Register, Fig. 18, comprises 4-Am 2917 cascaded units. Each Am 2917 is a 4-bit three-state bus transceiver. When 16-bit data is set up at its "A" inputs and a control signal (MBO) is applied at DRCP from the P. L. Reg., it stores the data in a set of D flip-flops. Another control signal (TRQ) enables the three-state outputs of the flip flops and the data is transmitted on the 16-data lines. In the receiving mode, the signal " BR " (Receiver Enable), applied at RLE input, clocks data from the 16-data lines into another set of D flip flops. The control signal MBI (from P. L. Reg.) enables the tri-state buffers at the output of those flip flops and places the received data on the D-Bus.

D-Bus. Just like C-Bus, the D-bus is effectively a multiplexer. It selects one of the five inputs to be applied to the Arithmetic and Logic Unit (ALU). All inputs are 16-bits wide and tied together through tri-state buffers.

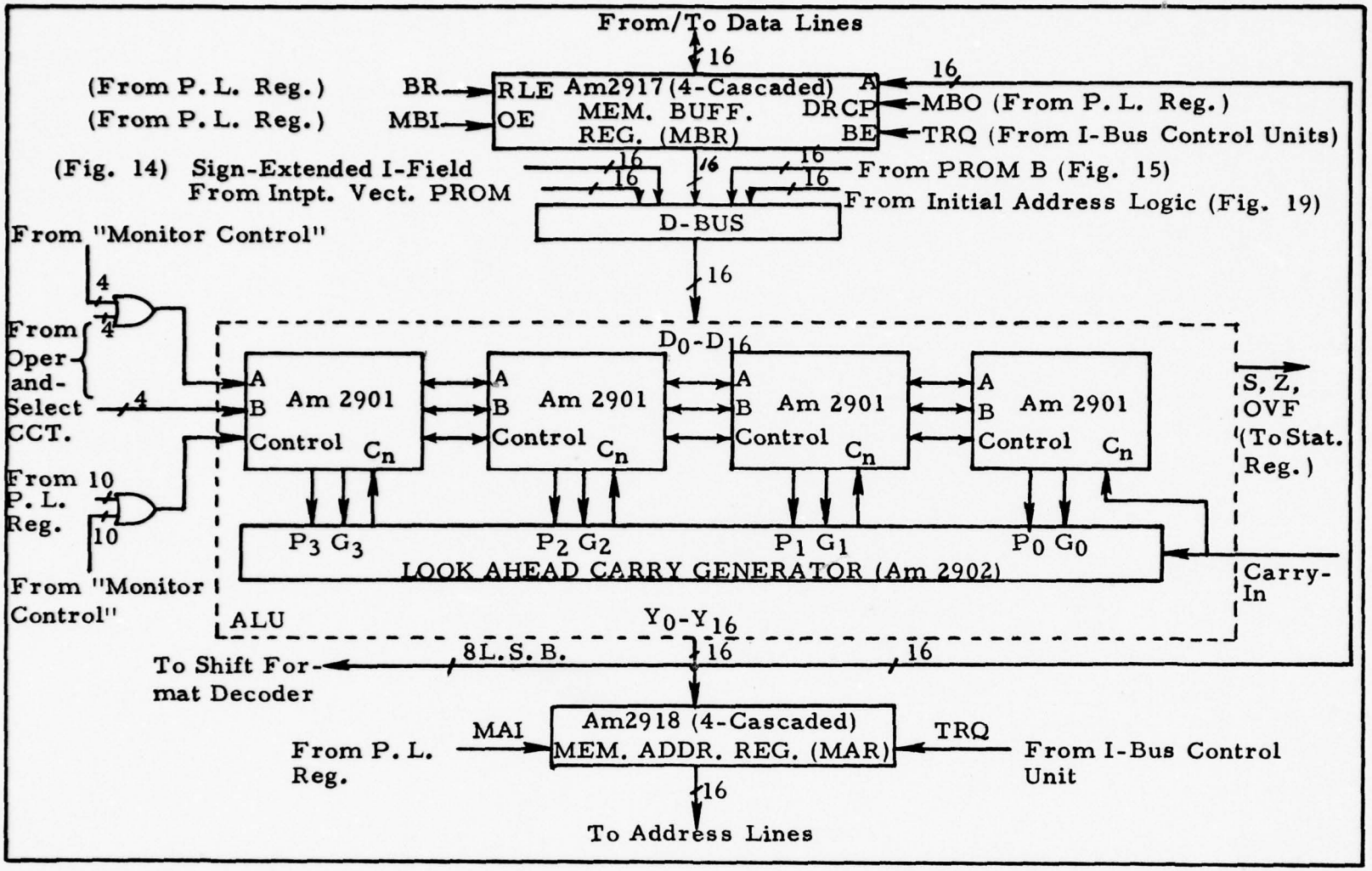


Fig. 18. Mem. Buff. Reg., D-Bus, ALU and Mem. Addr. Reg.

The 16-bit field from the "Initial Address Logic" (Ref Fig. 18)is used to provide the initial address of 0100 (Hex) when the processor is cleared/reset. The program counter is loaded with this address to start the "Power-up" phase. Figure 19 shows the detailed arrangement for this purpose.

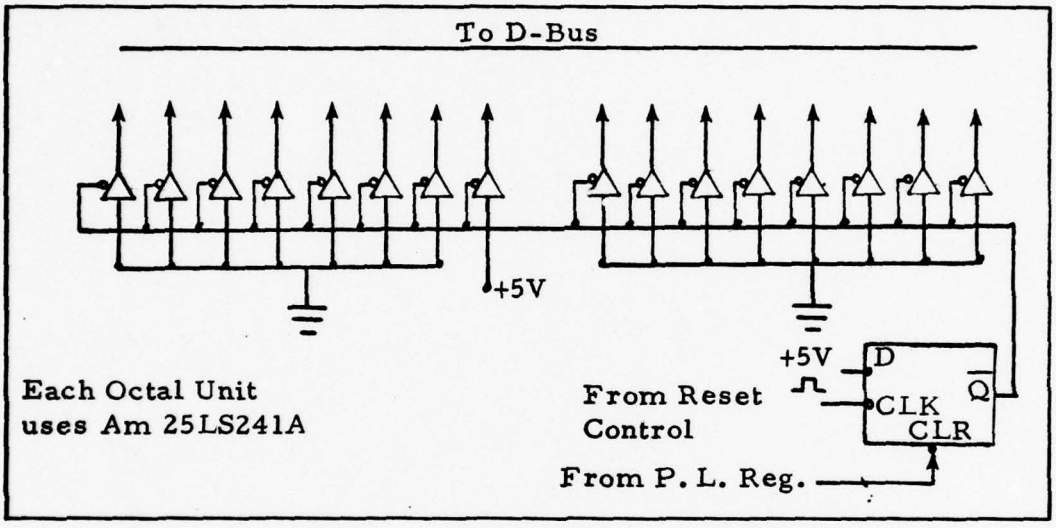


Fig. 19. Initial Address Logic

Arithmetic and Logic Unit (ALU)

The ALU consists of four Am 2901 CPU slices cascaded. It uses one Am 2902 as look-ahead carry generator. Figure 18 shows the circuit layout. The internal working and architecture of Am 2901 has already been discussed in the beginning of this chapter. The ALU receives 16-bit data from the D-Bus. Two 4-bit fields are applied at "A" and "B" inputs. These fields represent the address of a word to be read from/stored into the internal RAM of the ALU. The "B" address word is generated from the Operand Select Circuit. The "A"

address word may come from the same circuit or the Monitor Control Unit (to be described in Chapter V). The ALU needs three 3-bit fields to describe the Source of Operands, the function to be performed, and the destination to store the result of a subsequent computation. Another control bit enables/disenables the tri-state "Y" outputs. In the "Monitor" mode, these 10-bits are provided by the Monitor Control Unit.

Tables X, XI, and XII, list the details of the ALU source, function, and the destination fields and their associated mnemonics. These three control fields are provided from the P. L. Reg.

Table X
ALU Source Control Field

Micro Code			ALU Source Operands		Mnemonic
I_2	I_1	I_0	R	S	
0	0	0	A	Q	AQ
0	0	1	A	B	AB
0	1	0	0	Q	0Q
0	1	1	0	B	0B
1	0	0	0	A	0A
1	0	1	D	A	DA
1	1	0	D	Q	DQ
1	1	1	D	0	D0

Table XI
ALU Function Control Field

Micro Code I_5	I_4	I_3	ALU Function	Mnemonic
0	0	0	R Plus S	PLS
0	0	1	S Minus R	MIN
0	1	0	R Minus S	MINS
0	1	0	R Or S	OR
1	0	0	R And S	AND
1	0	1	$\overline{R}$ And S	MSK
1	1	0	R Ex-Or S	EOR
1	1	1	R Ex-Nor S	ENR

Table XII
ALU Destination Control Field

Micro Code I_8	I_7	I_6	Y-Output	Mnemonic	Explanation
0	0	0	F	Q	Result appears at "Y" and also stored in Q.
0	0	1	F	F	Result appears at "Y".
0	1	0	A	RA	A-port data appears at "Y," result stored in RAM.
0	1	1	F	RM	Result stored in RAM.
1	0	0	F	QRR	Result shifted right and stored in RAM and Q.
1	0	1	F	RR	Result shifted right and stored in RAM.

Table XII (continued)

Micro Code I_8	I_7	I_6	Y-Output	Mnemonic	Explanation
1	1	0	F	QRL	Result shifted left and stored in RAM and Q.
1	1	1	F	RL	Result shifted left and stored in RAM.

The bus oriented tri-state Y-outputs of ALU are directly connected to the Memory Address Register, Shift Formac Decoding Circuit, Processor Status Register, and Memory Buffer Register.

Memory Address Register (MAR). The Memory Address Register clocks ALU output data into 16-D flip flops on applying a Control Signal (MAI) from the P. L. Register. Another Control Signal (TRQ), from the I-Bus Control Unit, enables the three-state buffers at the outputs of the D-flip flops. This causes the contents of MAR to be placed on the address lines. The MAR is implemented by cascading two octal D-type registers (Am 25LS374).

Operand-Select Circuit. As indicated in Fig. 18, the A and B address words (4-bit each) for the ALU are generated from the Operand-Select Circuit for normal operation. Figure 20 shows the schematic for that circuit. It consists of a 4-bit UP/DOWN counter, two 4-input multiplexers, and two 4-bit latches.

The counter (R-counter) is parallel-loaded with R-field by the control signal LRC. Another two bits from the P. L. Reg. provide

the clock and the count up/count down controls as shown in Table XIII.

Table XIII
R-Counter Control Field

Micro Code I_1	I_0	Mnemonic	Explanation
0	1	DRC	Decrement R-Counter
1	1	ICR	Increment R-Counter
0	0	x	x x
1	0	LRC	Load R-Counter

Multiplexers A and B select one of the 4 inputs as shown in Fig. 20. The Multiplexer output is stored in the associated 4-bit latch. This configuration gives the flexibility of selecting the A and B address words for ALU from any of the four sources as required for a microinstruction.

Table XIV, specifies the operation of Multiplexers A and B.

Table XIV
Control Field for Multiplexers A and B

Micro Code I_1	I_0	Mnemonic	Explanation
0	0	LDP	Select 4-bits from P. L. Reg.
0	1	LDT	Select 4-bits from T-field.
1	0	LDR	Select 4-bits from R-field.
1	1	LDC	Select 4-bits from R-counter output.

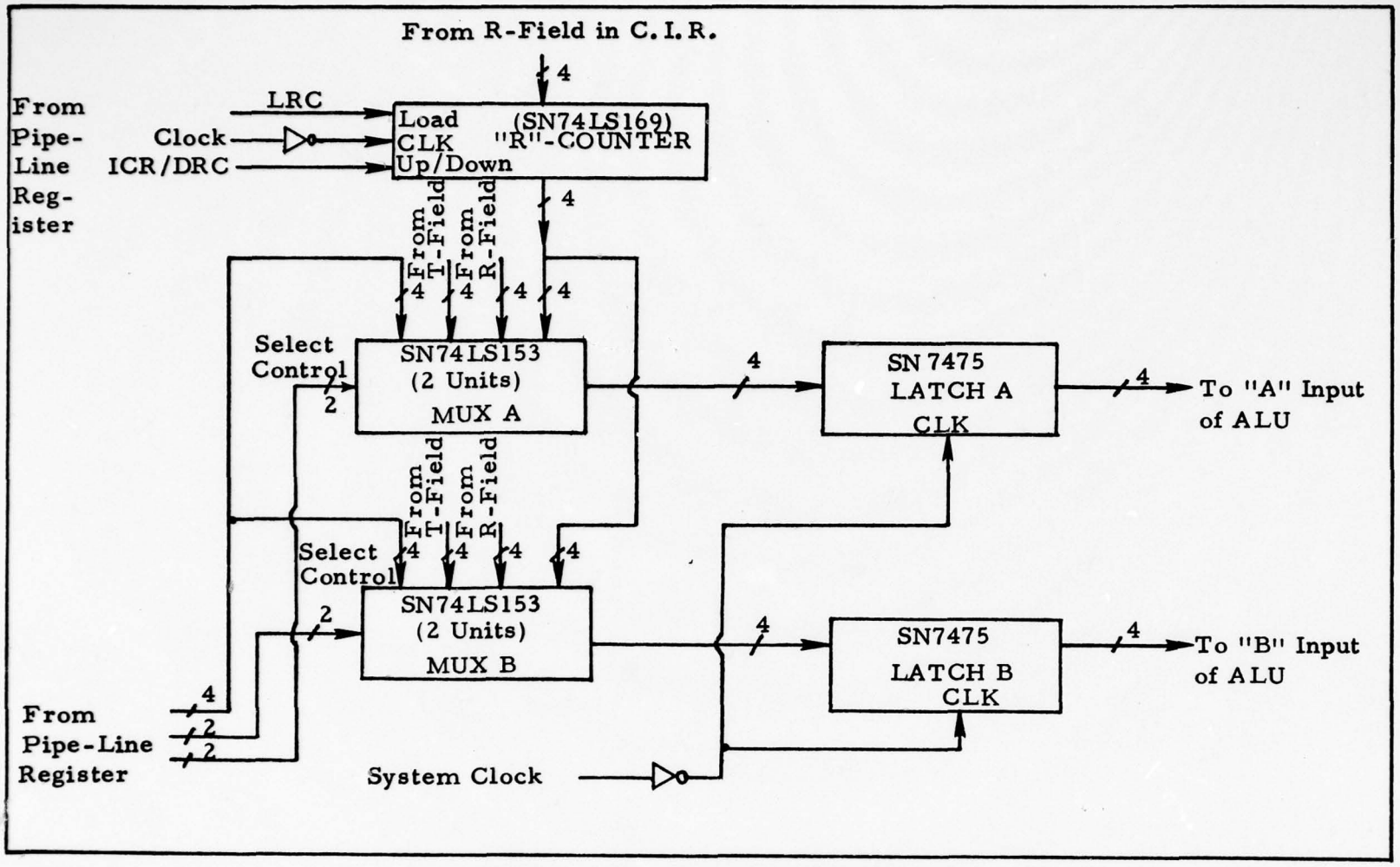


Fig. 20. Operand Select Circuit

Hardware for Shift Instructions

The Standard Format Instruction Set uses a special format for shift instructions. Eight LSB in a register (in ALU RAM) specify the various informations about the shift instructions. Five LSB contain the shift count and the three MSB indicate the nature (Arithmetic or Logic) and direction (right, left or rotate) of the shift operation. The MSB of OP-Code C1 decides whether it is a single shift or double shift operation. It is, thus, necessary to decode the shift format before effecting the desired shift operation.

The Shift Format Decoder, as shown in Fig. 21, achieves this purpose. The Shift Buffer Register receives 8-bit format from the ALU output. It routes 5-LSB to the loop counter in the Microprogram Controller and 3-MSB to PROM C.

The PROM is enabled by the control signal $\overline{\text{VECT}}$ from the Am 2910 chip. It puts out a 12-bit starting address for each shift instruction.

After decoding the shift format, the specific shift operation is accomplished with the help of right-shift and left-shift multiplexers associated with ALU. Figure 22 shows the details for this configuration.

Shift linkages indicated in Figure 22 take into account all possible shift operations. "SIGN" is provided by the status register. It is needed for Arithmetic shifts. For rotations, LSB of RAM (LR), MSB of RAM (MR), LSB of Q Reg. (LQ), and MSB of Q-Reg. (MQ),

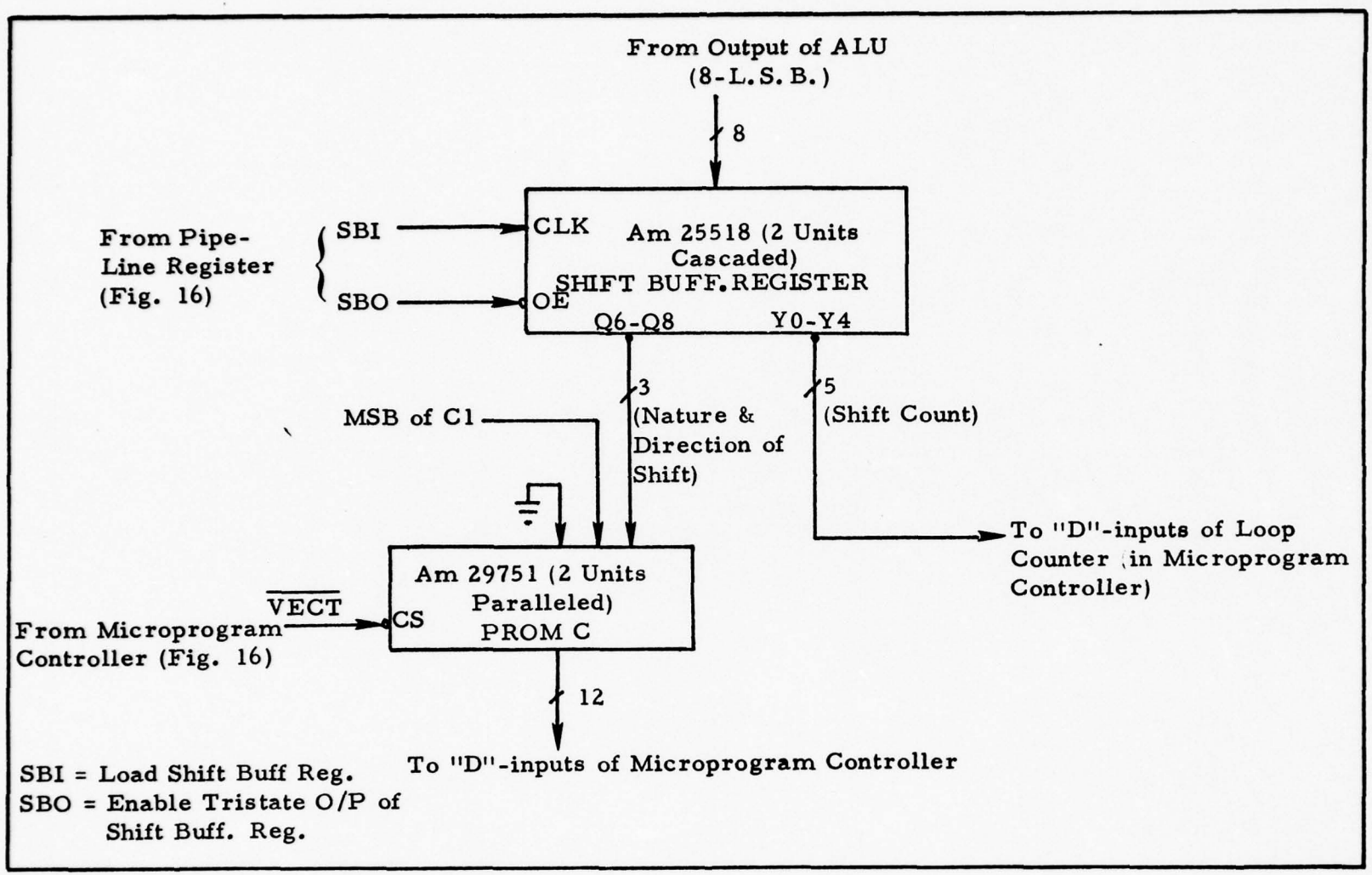


Fig. 21. Shift Format Decoder

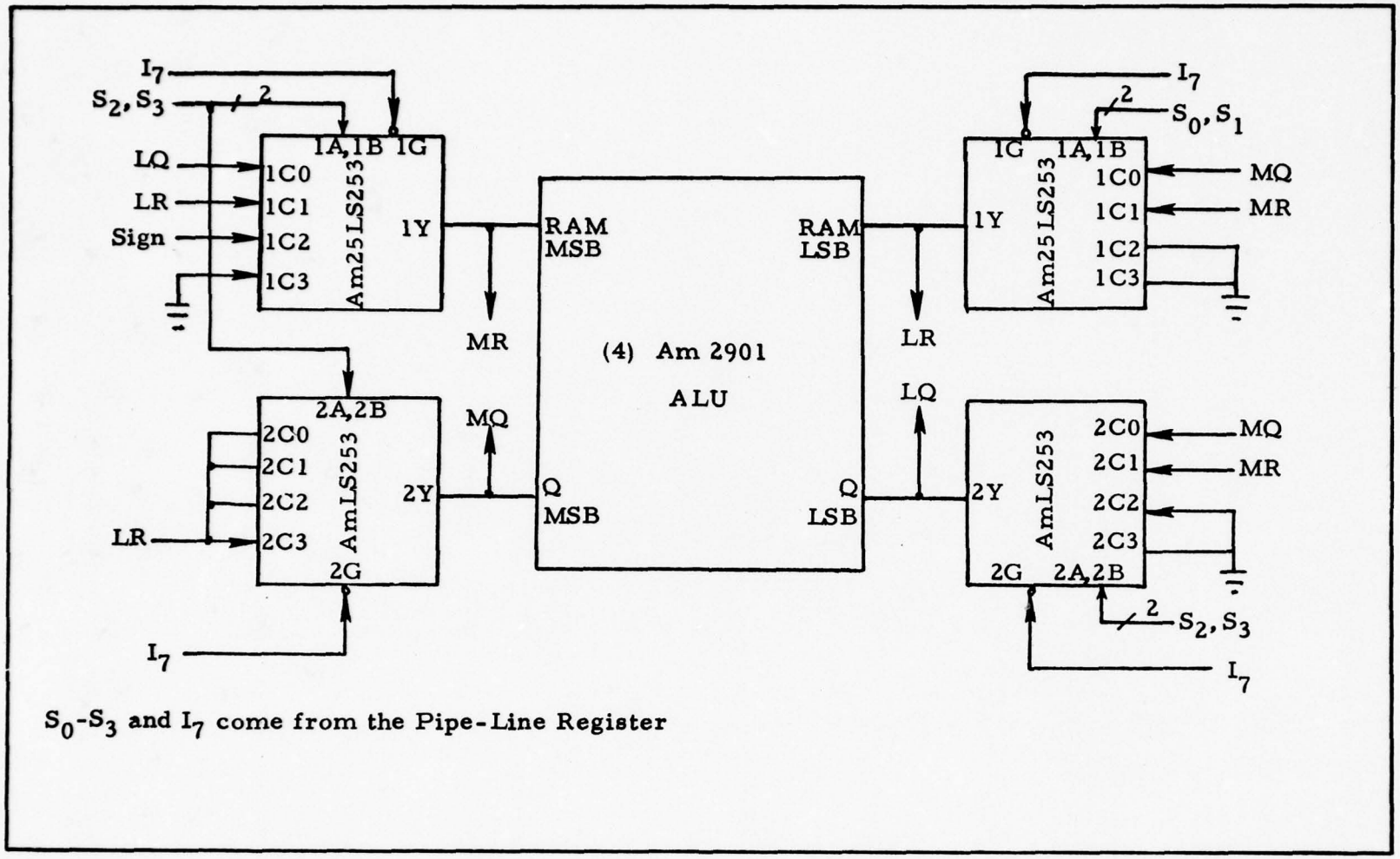


Fig. 22. Linkages for Shift Instructions

can be appropriately connected.

A 4-bit field (S0-S4) along with the ALU destination control code (described in Table XII), controls all the shift operations as given in Table XV.

Table XV
Shift Control Field

Micro Code							Mnemonic	Explanation
S_3	S_2	S_1	S_0	I_8	I_7	I_6		
1	1	0	0	1	0	1	LRS	Logical shift right single
1	1	0	0	1	0	0	LRD	Logical shift right double
1	0	0	0	1	0	1	ARS	Arithmetic shift right single
1	0	0	0	1	0	0	ARD	Arithmetic shift right double
0	1	0	0	1	0	1	RRS	Rotate right single
0	0	0	0	1	0	0	RRD	Rotate right double
0	0	0	0	1	1	0	LLD	Logical shift left double
0	0	1	1	1	1	1	LLS	Logical shift left single
0	0	1	1	1	1	1	ALS	Arithmetic shift left single
0	0	0	1	1	1	1	RLS	Rotate left single
0	1	0	0	1	1	0	RLD	Rotate left double

Processor Status Register. The Processor Status consists of two components; the internal status (sign, zero, OVF), and the interrupt mask for entire system. Fig. 23 shows the requisite layout. The internal status is available from the ALU. Sign, Zero, and Overflow

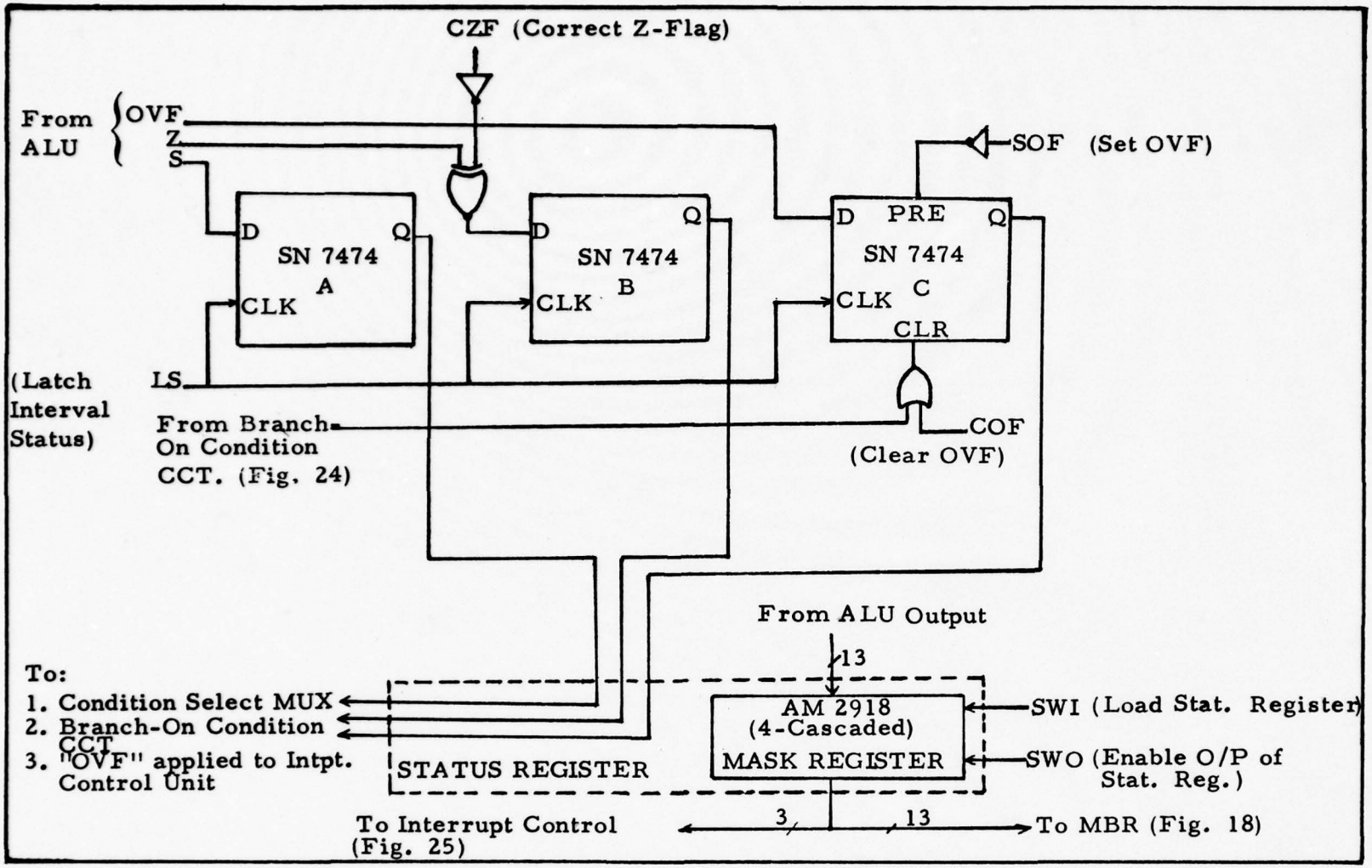


Fig. 23. Status Word Register

information is stored in the three Flip-Flops at the end of each microinstruction and fed to the Condition-Select Multiplexer. The Flip-Flops A and B together are called the "Condition Code Register (CCR)." Whenever a branch is made on Overflow Condition, the Flip-Flop C is cleared.

The System Mask is a 13-bit information. Three MSB denote overflow mask, device error mask, and the memory error mask. These three bits are applied to the Interrupt Control Unit. The ten LSB contain the mask information for the Bus Interface Unit. The Mask Register is loaded under the program control.

Hardware for "Branch-on Condition." Branch-on Condition instruction checks for a specific value of the R-field against a condition given by the contents of the CCR (Sign and Zero Flags). If both are true, the program control is transferred to the Derived Address. Otherwise, the next sequential instruction is executed.

Table XVI lists the details of various conditions and Fig. 24 shows the logic circuit to realize the Branch-on-Condition function. The output of the circuit is applied to the Condition-Select Multiplexer.

Interrupt Control Unit

The Interrupt Control Unit uses one Am 2914 chip. The processor is responsible for handling interrupts due to Illegal OP-Code, Overflow, Device Error, and Memory Error. These interrupt

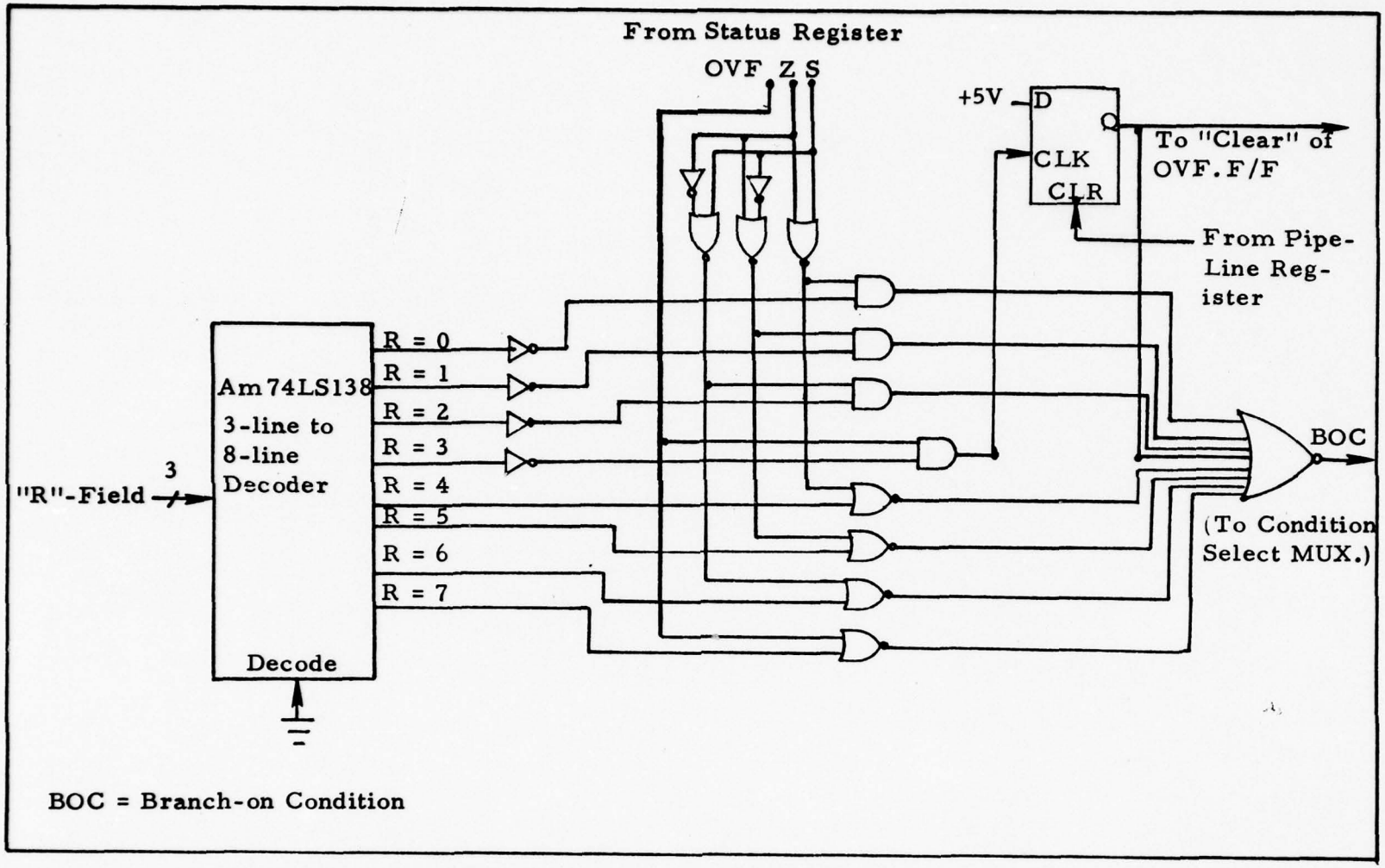


Fig. 24. Branch-on Condition Logic

Table XVI
Tabulation of "Branch-on Condition" Function

R-Field	Condition	Action
0 0 0	CCR. EQ. 00	DA ⟶ PC
0 0 1	CCR. EQ. 01	DA ⟶ PC
0 1 0	CCR. EQ. 10	DA ⟶ PC
0 1 1	OVERFLOW	DA ⟶ PC
1 0 0	CCR. NE. 00	DA ⟶ PC
1 0 1	CCR. NE. 01	DA ⟶ PC
1 1 0	CCR. NE. 10	DA ⟶ PC
1 1 1	NO OVERFLOW	DA ⟶ PC

inputs and their respective mask bits are applied to the Interrupt Control Unit as shown in Fig. 25. An Illegal OP-Code causes a non-maskable interrupt. Hence, its Mask Bit is permanently set in Logic One state.

Whenever an interrupt input goes LOW, it is compared with its mask bit. If mask bit is set, an Interrupt Request is generated and a 3-bit vector is put out to VECTOR MAP. PROM. The Interrupt Request is one of the inputs to the Condition-Select Mux.

A 4-bit field controls all the functions of the Interrupt Control Unit. This field is effective only if the Enable Signal is LOW.

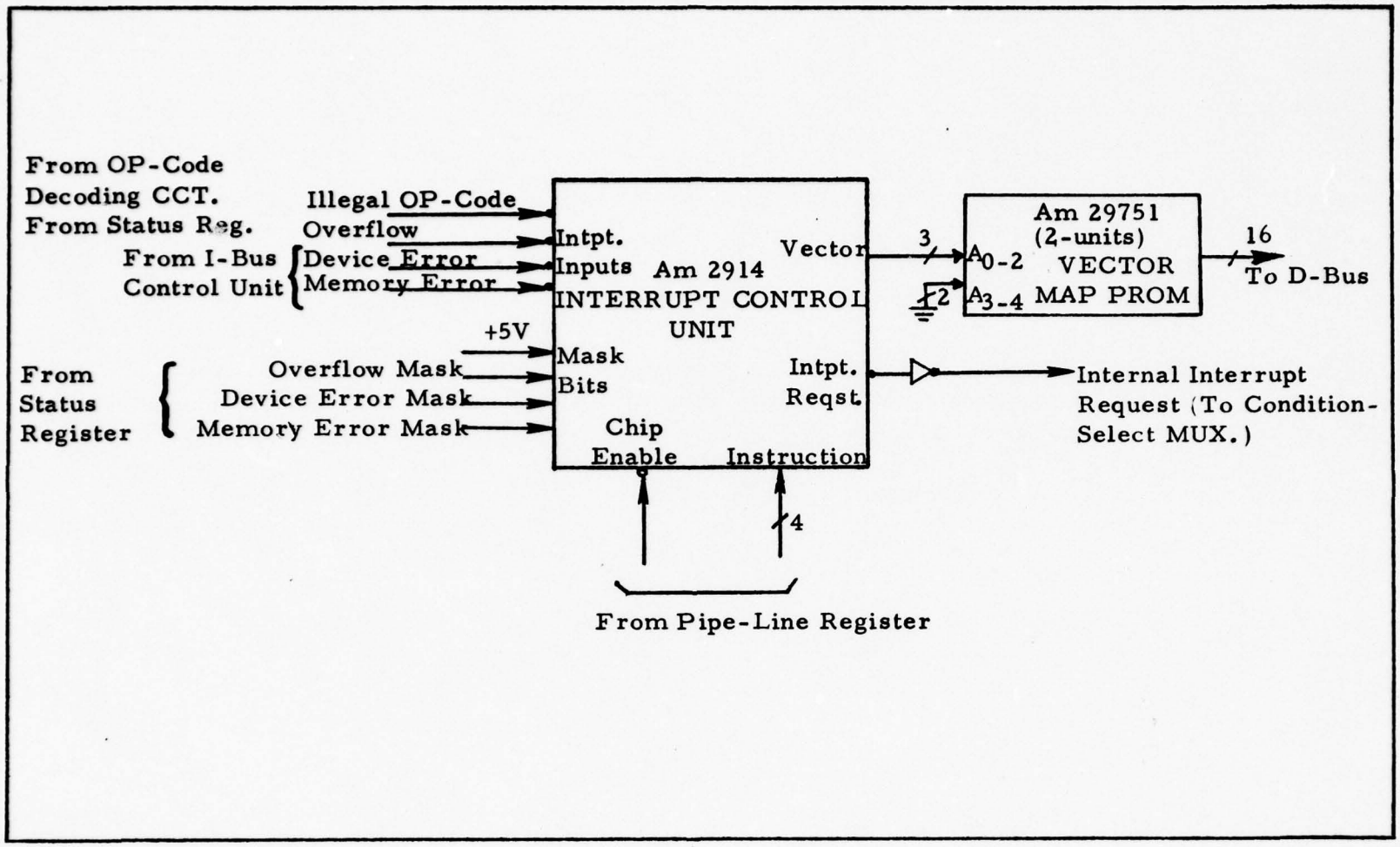


Fig. 25. Interrupt Control

This circuit caters for the interrupts which are internal to the Processor. In the case of external interrupts, the interrupting device sets the "External Interrupt" flag at the input of the Condition Select MUX (Fig. 17). In response, the Processor generates "Interrupt Acknowledge (IAK)." The interrupting device, then, places 'Trap Vector' (TV) on the data lines. This is discussed in detail in the "I-Bus Control Unit."

The details of the 4-bit control field for the Interrupt Control Unit are given in Table XVII.

Table XVII
Interrupt Control Field

Micro Code				Mnemonic	Explanation
I_3	I_2	I_1	I_0		
0	0	0	0	MCL	Master clear.
0	0	0	1	CAI	Clear All Interrupts.
0	1	0	0	CIV	Clear Interrupt, Last vector read.
0	1	0	1	RVC	Read vector.
0	1	1	0	RSR	Read Status Register.
1	1	0	0	CMR	Clear Mask Register.
1	1	0	1	DIR	Disable Interrupt Request.
1	1	1	0	LMR	Load Mask Register.
1	1	1	1	EIR	Enable Interrupt Request.

I-Bus Control Unit

Figure 26 presents the detailed logic circuit for the I-Bus control unit. It enables the Processor to generate the necessary control signals for transmitting data to/receiving data from the Main Memory or I/O devices.

The signal BRQ sets Flip Flop D and if no other device is in control of the I-Bus, BGRI signal is LOW. This sets Flip Flop E, producing TRQ. A 4-bit Processor ID (0000) is generated and the BREL signal goes LOW which clears Flip Flop D. The TRQ enables DRCV and IOSL signals on to the control lines. The 16-bit address and 16-bit data is also simultaneously placed on the address and data lines. DRCV, IOSL, Address and Data are required to be set up in their respective registers before initiating BRQ. The device whose address was transmitted from MAR, collects data and sends back TACK. This signal is delayed 150 nano-seconds to overcome the Bus-skew and then it clears Flip Flop E, removing TRQ, address and data. Flip Flop G generates TCP to indicate "Transfer Complete" to the Condition-Code MUX. During Receiving mode, 2-control bits from the P.L. Reg. and TACK are used to load the data from the 16-data lines either in MBR or in CIR. If Main Memory is addressed with a non-existent address, it generates TTO signal which removes TRQ as before and sets Flip Flop E to indicate Memory Error.

In the case of external interrupts, the I-Bus control unit generates IAK. The interrupting device puts Trap Vector on the data

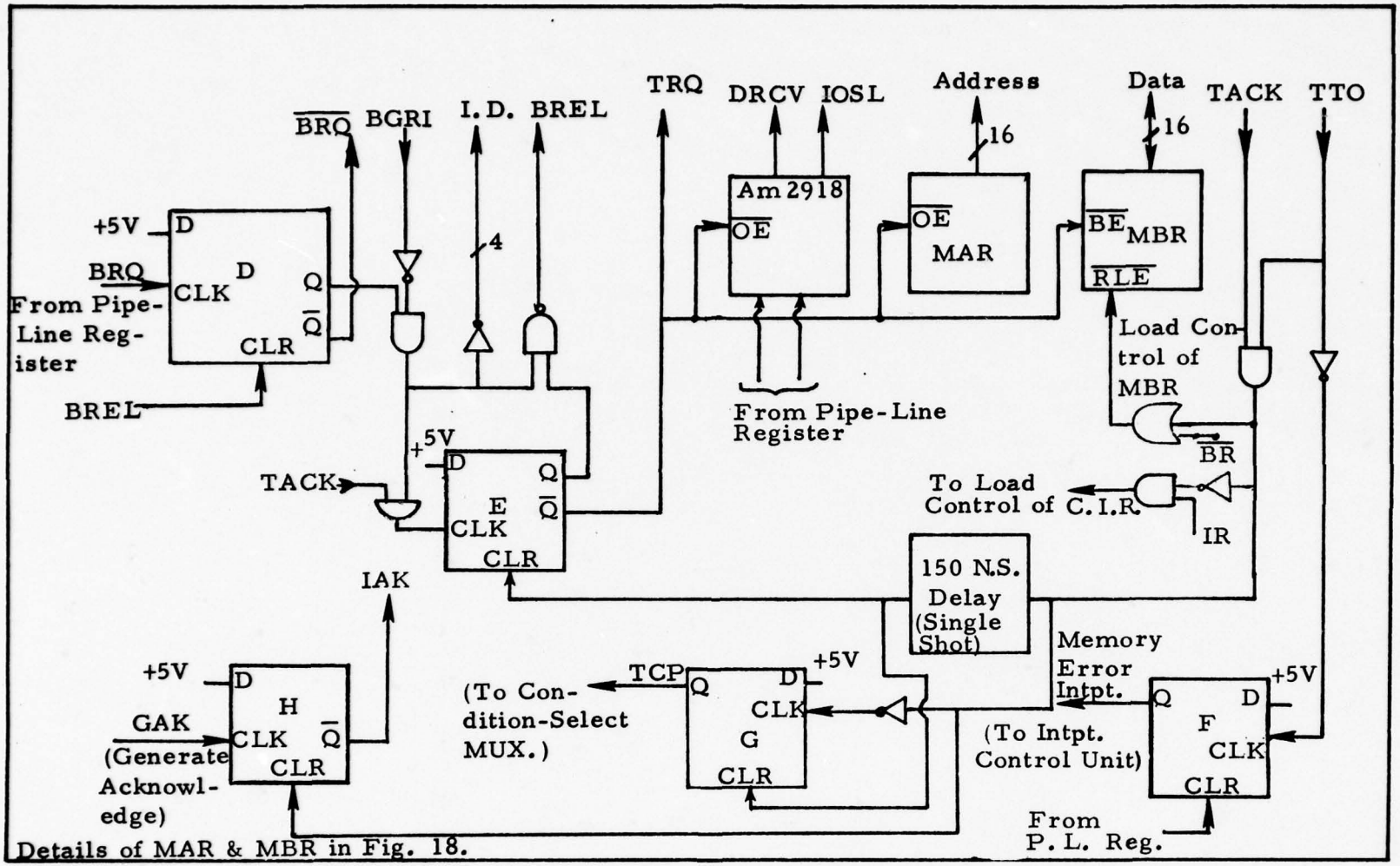


Fig. 26. I-Bus Control Unit

lines from where it is loaded in MBR.

Summary

In this chapter, the prominent members of the Am 2900 family have been described. The processor implementation has been presented in detail. The discussion of each unit in the processor includes the description of the associated control field as well. All the control fields are regrouped and tabulated again in Appendix D for quick reference.

The next chapter attempts to present the "Microprogramming" aspect of the DP/M Processor as designed in this chapter.

IV. Microprogramming

This chapter describes the Microprogramming Philosophy for the various Machine-States of the Processor implemented in Chapter III. First, a brief description of each state is given, which is followed by the discussion of the Microinstruction Format. Lastly, the arrangement of the micro-codes has been outlined for quick reference.

State Transition Diagram

The State Transition Diagram for the Processor is shown in Fig. 27.

The first state represents the power up phase in the processor cycle. Here, the processor clears all flags, Interrupt Stack Pointer (ISP), Status Register, and loads the starting address 0100 (HEX) in the Program Counter Register (PC). It, then, checks for the "HALT" condition before going into the second state. If the HALT switch, on the control panel, is depressed, the processor is put in a waiting loop. In the first state all functions except the starting address generation, are controlled through the micro-codes.

In the second state, the contents of PC are loaded into MAR and I-Bus requested to fetch Macroinstruction from the Main Memory. The instruction is loaded into CIR. All the functions in this state are controlled through the micro-codes.

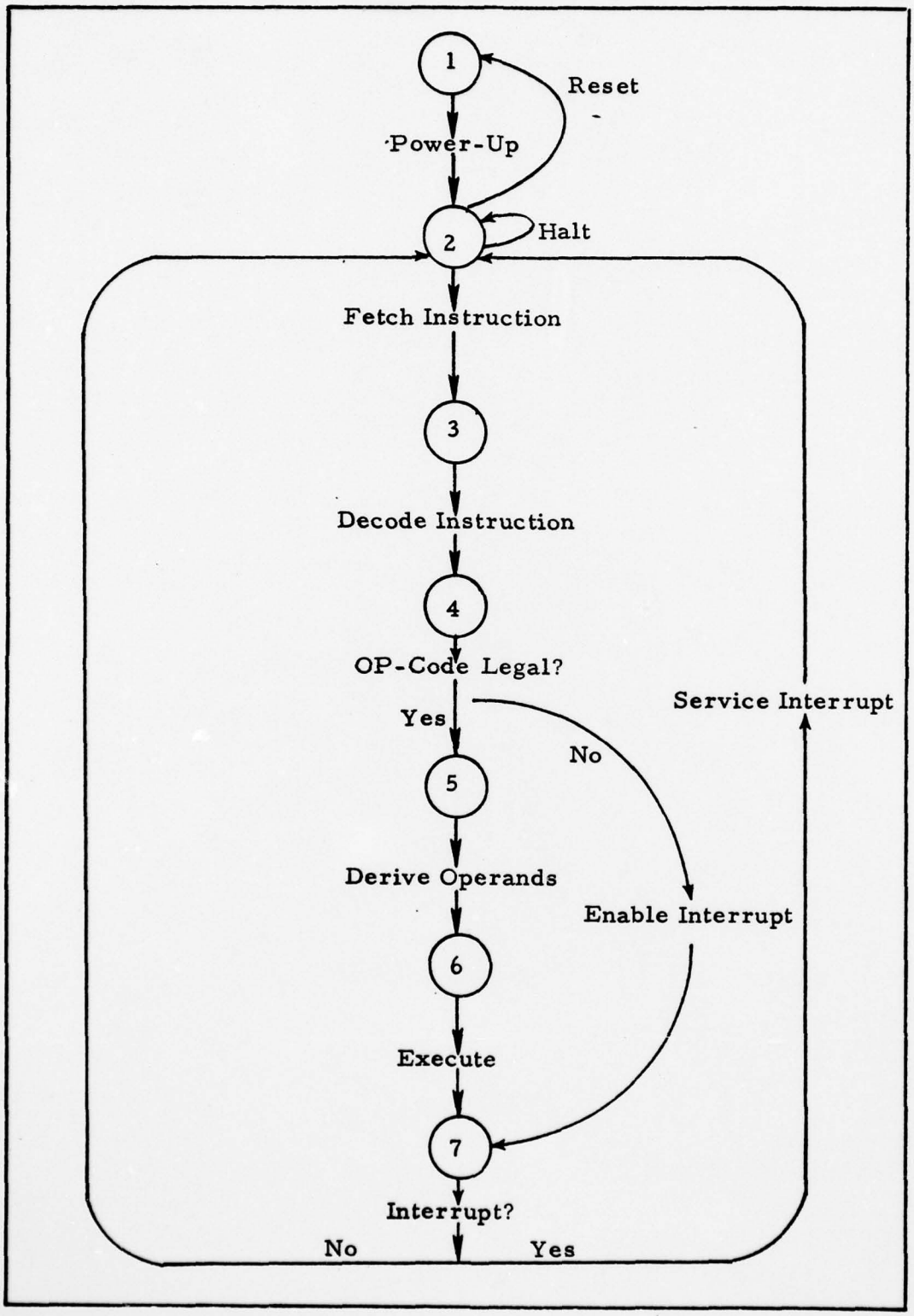


Fig. 27. State Transition Diagram

The third state uses hardware (Ref Fig. 13, Chapter II) for decoding various fields in the instruction to compute the Addressing Mode and the OP-Code for subsequent operations.

In the fourth state, the legality of the OP-Code is checked. If it is illegal, the micro-codes enable an unmasked interrupt and the processor branches to state seven. If the OP-Code is found legal, the control passes on to state five.

The state five represents the Operand Derivation Phase. The Operand Derivation depends on the Addressing Mode as decoded in state three. For the Register Indirect Autoincrement Mode (M = 10, T = 7), Direct Mode (M = 11, T = 0), and Direct Indexed Mode (M = 11, T $\neq$ 0), an additional memory-access is required to fetch the second 16-bit word before computing the operand. All the Addressing Modes have been described in Chapter I.

The state six is the Execution Phase of the Macroinstruction. Each OP-Code, decoded in state three, produces a unique sequence of microinstructions which manipulate the operands to perform the desired operation. The computed result is stored either in Main Memory or in one of the registers located in the file stack (RAM) of ALU. The RTL description of the given instruction sets is contained in Appendix A.

In the last state, a check is made for any internal or external interrupt. If an interrupt is detected, the processor executes the Interrupt Handling Routine, services the interrupt and then enters

state two to repeat the cycle. If no interrupt is found, the control directly passes on to state two.

Microinstruction Format

For Microprogramming the Processor, various signals are needed to control the functions of ALU, Microprogram Controller, Multiplexers, and the Registers. The Microinstruction Format, shown in Fig. 28, embodies all the requisite control signals.

It is a 64-bit word which consists of 15-control fields. The description of each field is given below.

ALU Control. This field comprises 3 microfields, one each for designating the source(S) of operands, Function (F) to be performed, and the Destination (D) of the computed results. The details of S, F, and D are given in Tables X, XI, and XII (Chapter III) respectively.

Carry Control (Cn). It is a single bit used to set the carry-in (Cn) to either "1" or "0" as required by the ALU operation.

Multiplexer Control. The Multiplexer Control consists of 3-control fields, one each for Multiplexer A (MUX.A), Multiplexer B (MUX.B), and Multiplexer C (MUX.C). Each field uses 2-bits. The details for MUX.A and MUX.B are shown in Table XIV, and MUX.C control is given in Table VII (Chapter III).

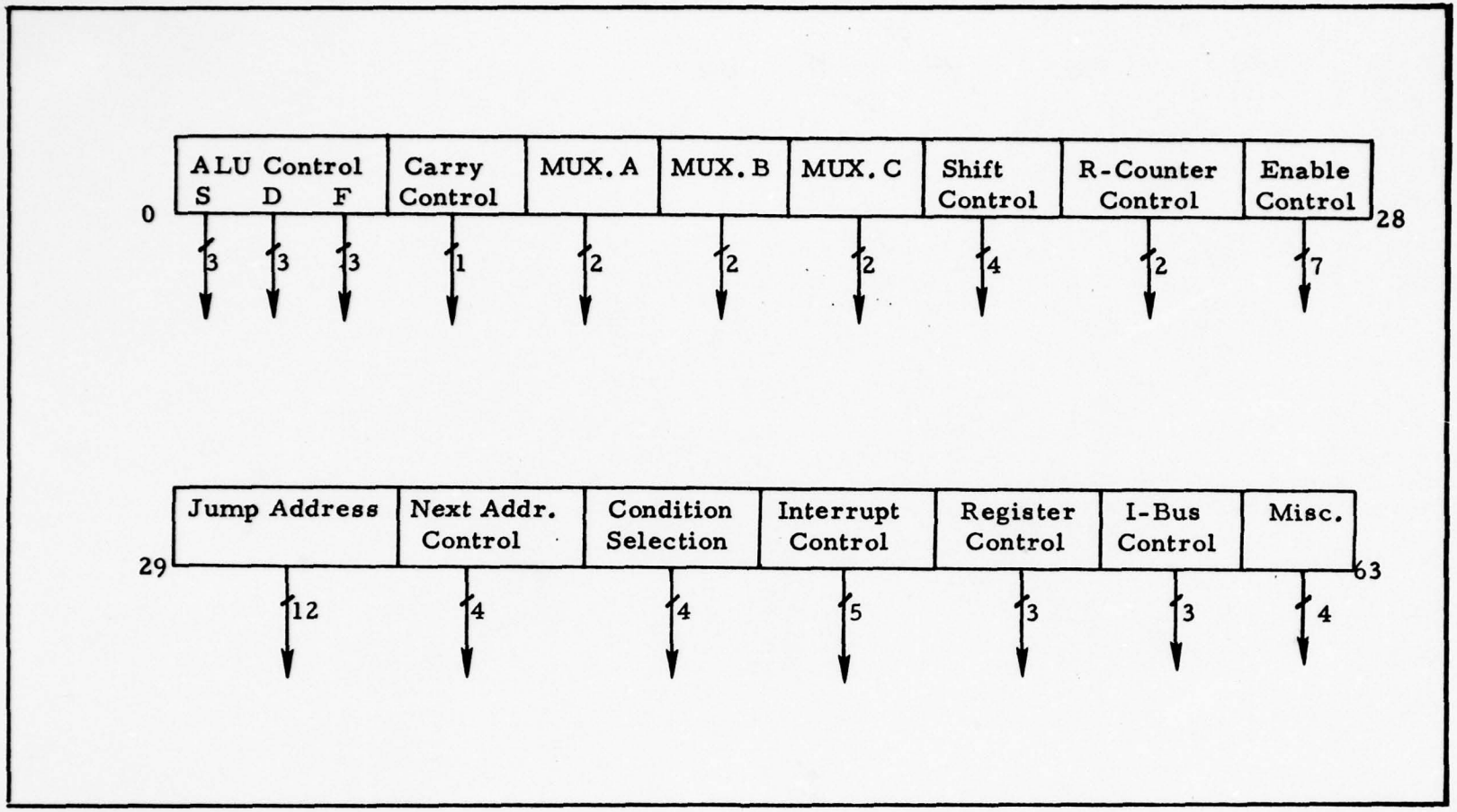


Fig. 28. Microinstruction Format

Shift Control. The Shift Control is a 4-bit field. It generates appropriate shift linkages to effect a shift operation. It is used in conjunction with the Destination Control of the ALU. The details are tabulated in Table XV (Chapter III).

R-Counter Control. It is a 2-bit field used to increment/decrement the R-Counter as given in Table XIII (Chapter III).

Enable Control. The Enable Control provides command signals to control the output of various elements in the processor. It also decides whether to load the data (from the 16-data lines) into CIR or MBR whenever new data is to be read from the Main Memory. It is a 7-bit field, controlling seven functions which are not mutually exclusive. The details are given in Table XVIII on the next page.

Jump Address. It is a 12-bit field which determines the location in the Micromemory to which a jump is made if required by a microinstruction.

Next Address Control. This 4-bit field controls the operation of the Microprogram Controller as discussed in Chapter III. The details of the micro codes and the mnemonics are shown in Table VIII (Chapter III).

Condition Selection. The Condition Selection field provides command signals for the Condition Select Multiplexer. It consists of 4-bits. The details of various micro codes are available in Table IX in Chapter III.

Table XVIII
"Enable" Control Field

Micro Codes							Mnemonic	Explanation
I_6	I_5	I_4	I_3	I_2	I_1	I_0		
0	0	0	0	0	0	1	MP	Enable output of microprogram controller.
0	0	0	0	0	1	0	AL	Enable output of ALU.
0	0	0	0	1	0	0	ET	Enable T-field at loop counter I/P.
0	0	0	1	0	0	0	PB	Enable PROM B.
0	0	1	0	0	0	0	LC	Load Condition Code Register.
0	1	0	0	0	0	0	BR	Load data in MBR.
1	0	0	0	0	0	0	IR	Load data in CIR.

Interrupt Control (Intpt. Control). The 5-bit Interrupt Control field provides signals to the Interrupt Control Unit (Am 2914) for enabling/disabling the interrupts, loading/clearing the Mask, and for producing the Trap Vector as outlined in Table XVII in Chapter III.

Register Control. The 3-bit Register Control field functions as shown in Table XIX.

Table XIX
Register Control Field

Micro Code I_2	I_1	I_0	Mnemonic	Explanation
0	0	0	x x	x x
0	0	1	MAI	Load Address in MAR.
0	1	0	MBI	Put data on D-Bus (from MBR).
0	1	1	MBO	Put data in MBR for transmission.
1	0	0	LIR	Load I-Field in I-Register.
1	0	1	SIO	Put sign extended I-Field on D-Bus.
1	1	0	SBI	Load data into Shift Buffer Register.
1	1	1	SBO	Put data out from Shift Buff. Register.

I-Bus Control. The I-Bus Control field consists of three bits, BRQ, DRCV, and IOSL. They are used to get control of the I-Bus and establish communication between the processor and the other devices.

BRQ, when "1," represents Bus Request. DRVC, if "1," means data is to be transferred from an external device (like Main Memory) to the processor. If DRVC is "0," then the processor sends data to a device. IOSL, when "1," means the external device being addressed is Main Memory. IOSL, when "0," stands for any other device except the Main Memory (like BIU).

Miscellaneous Control Field (Misc.). This 4-bit field provides various control signals as given in Table XX.

Table XX
"Miscellaneous" Control Field

Micro Code				Mnemonic	Explanation
I_3	I_2	I_1	I_0		
0	0	0	0	x x	x x
0	0	0	1	CZF	Correct "Z"-flag.
0	0	1	0	$\overline{\text{RLD}}$	Load Loop Counter.
0	0	1	1	SOF	Set Over Flow flag.
0	1	0	0	GAK	Generate Acknowledge.
0	1	0	1	LVE	Load Vector.
0	1	1	0	SWI	Load Status Word in Status Word Reg.
0	1	1	1	SWO	Enable Output of Status Word Reg.
1	0	0	0	CSW	Clear Status Word.
1	0	0	1	PQS	Set Product/Quotient Flip Flop.
1	0	1	0	PQC	Clear Product/Quotient Flip Flop.
1	0	1	1	COF	Clear Over Flow

Arrangement of Flow Charts and Micro Codes

The RTL flow charts were drawn and the micro-codes were written for the specified instruction set. The flow charts appear in Appendix B and the micro codes are given in Appendix C. For ease of reference, the flow charts and the micro codes have been arranged

in the following sequence:

a. Power-up Phase

b. Fetch Phase

c. Operand Derivation Phase

d. Execution Phase

e. Interrupt Handling Phase.

All these phases have been described earlier in this chapter. An attempt has been made to keep the RTL flow charts directly related to their corresponding micro codes.

The codes for each microinstruction appear on two consecutive pages (due to long Microinstruction Format) facing each other. Both the pages carry the address of the microinstruction. It helps to keep the continuity from one page to the other. At the end of each micro-instruction, brief remarks are given to highlight the functional description of that microinstruction.

Summary

This chapter has presented the description of the State Transtion Diagram and the Microinstruction Format for the Processor designed in Chapter III. The layout of the RTL flow charts and the micro-code has also been explained.

The next chapter describes the function and design of a Micro-Level Monitor.

V. Monitor Control Unit

This chapter describes the function and the design of a Micro-Level Monitor Control Unit. It would be used to monitor the execution of microinstructions when the processor operated in "Manual Mode." It is assumed that the various switches needed for monitoring purposes would be available on the maintenance/control panel of the DP/M Processor.

Function. The Monitor Control Unit is aimed to be a help in debugging the micro-code by displaying the contents of various registers when a microinstruction is executed. The registers monitored are: Current Instruction Register, Pipe Line Register, Status Register, and the Register File in the RAM of ALU. The output of the Microprogram Controller can also be monitored. Additionally, any 12-bit address can be selectively applied to the Microprogram Memory to load a specific microinstruction in the Pipe Line Register.

Monitoring the current Instruction Register, can help in knowing the contents of X, C1, M, T, and R fields in the macroinstruction. M and T fields give the indication of a specific addressing mode. One can follow through the sequence of microinstructions to check the address/operand derivation phase. In this respect, appropriate micro-code/flow charts can be used to verify the results. The address of a microinstruction is displayed by monitoring the output

of the Microprogram Controller. The contents of a microinstruction are displayed by monitoring the Pipe Line Register. Thus, by monitoring these two units, one can verify as to which microinstruction is executed and what control signals are generated. Similarly, by monitoring the Register File in ALU, one can verify the correct execution of a microinstruction.

This design, however, does not offer the capability of changing the contents of a microinstruction in the Micro Memory. This is due to using PROMs and not RAMs for realizing the Micro Memory. With RAM chips, it would be possible but the entire Memory would have to be reprogrammed every time the processor power was switched on.

In the Monitor mode, the system clock is inhibited from the processor and a "Manual Clock" takes over the operation.

Design of Monitor Control Module

The Monitor Control Module consists of four units which deal with General Purpose Register-File monitoring, Special Register monitoring, Direct Address Selection, and the Manual Clock Generation. These units are described in the subsequent paragraphs.

General Purpose Register-File Monitoring. The general purpose register-file consists of eight registers located in the RAM of ALU. In order to access the contents of any of these registers, the register address (e.g. $R_7 \Longrightarrow 0111$) is applied to one of the two

address ports of the ALU. Also, appropriate source, function, and destination controls are selected at the ALU control lines so as to select data from the RAM, perform some arithmetic/logic function, and place the result at the output of ALU.

In the present design, the address of the register (to be monitored) is applied to the 'A' address port of the ALU as shown in Fig. 29. The circuit generates micro codes for the 'S,' 'F,' and 'D' inputs of the ALU. Using these micro codes, the ALU performs an 'OR' function between 0 and the contents of the register (whose address appears at 'A' port of ALU) and places the result at the output of ALU.

The register to be monitored is selected at the input of an 8-line to 3-line encoder. To avoid the "Switch Debouncing," the 'GS' output of the encoder is delayed through the "single shot" to produce the clock pulse which latches the encoder output. The delay is adjustable and may be set to about 100 m.s. to avoid transients. The same pulse also controls the flip flops to generate the control signals for the source, destination, and the function selection. Simultaneously, another flip flop is pre-set to disable the P. L. Register and to enable the tri-state buffers connected to the ALU output. These buffers apply the ALU output to a hexadecimal display. After monitoring the contents of a register, all the flip flops are to be reset through "CLEAR" switch before monitoring any other register.

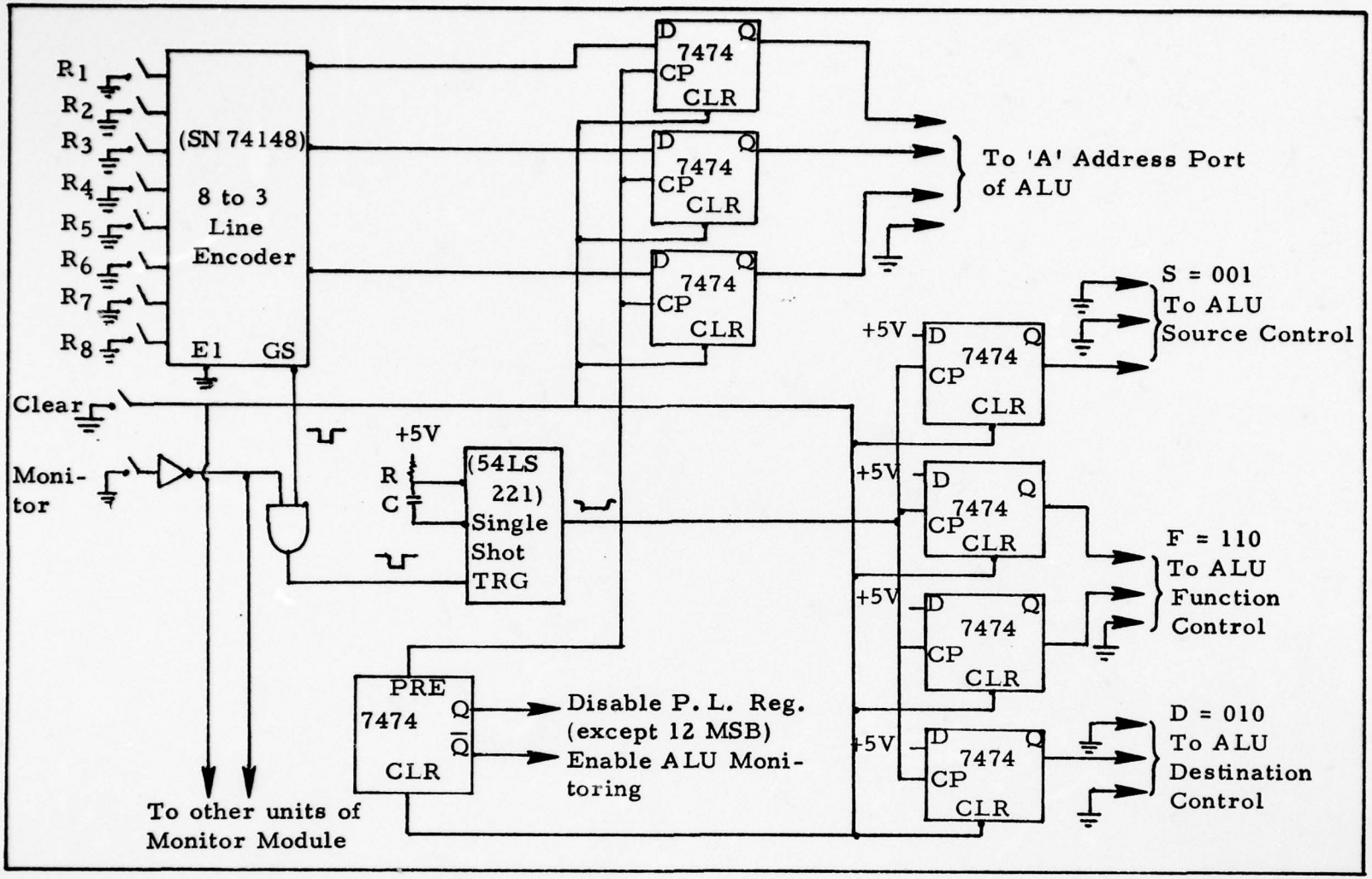
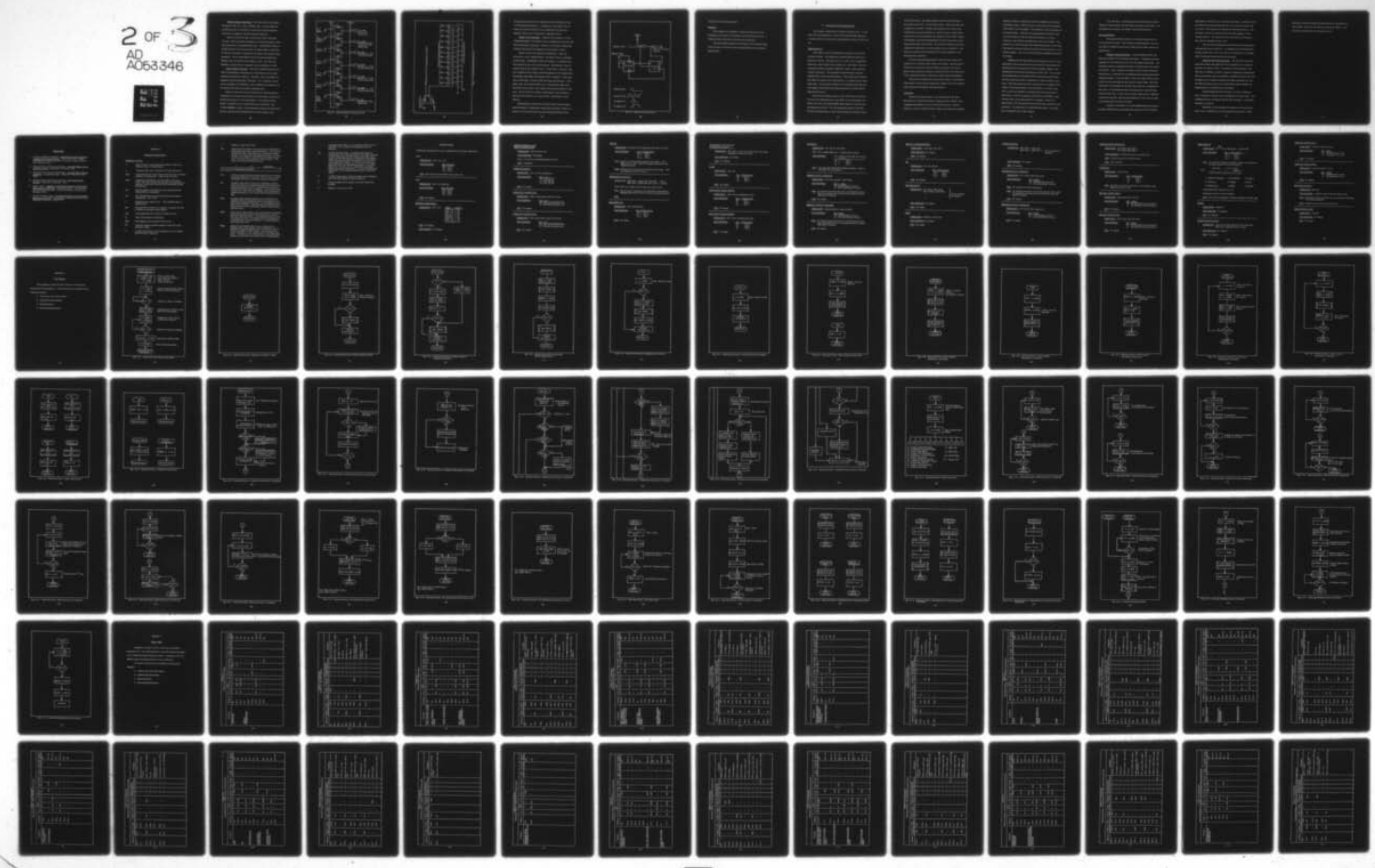


Fig. 29. Register-File Monitor Unit

AD-A053 346 AIR FORCE INST OF TECH WRIGHT-PATTERSON AFB OHIO SCH--ETC F/G 9/2
EMULATION OF THE PROCESSOR FOR DISTRIBUTED PROCESSOR/MEMORY SYS--ETC(U)
DEC 77 E MUHAMMAD

UNCLASSIFIED AFIT/GE/EE/77-31 NL

2 OF 3
AD
A053346



Special-register Monitoring. This unit controls the monitor function for CIR, P. L. Reg., and Stat. Reg. It also handles the monitoring of the 12-bit address produced by the Microprogram Controller and applied to the Microprogram Memory.

Figure 30 shows the logic circuit for the above mentioned sub-unit. Each of the special registers is selected through a switch which presets the corresponding flip flop. If "MONITOR" switch is already pressed at the control panel, an enable signal is generated for the tri-state buffers connected to the output of the register being monitored. All tri-state buffers drive the common hexadecimal display (same one used for ALU output as well). The Pipe Line Register is monitored as 4-segments, each consisting of 16-bits.

Direct Address Selection. The Direct Address Selection unit offers the flexibility of inserting any 12-bit address to access the contents from the Micro Memory. Normally, the 12-bit address is supplied by the Microprogram Controller. It is, therefore, necessary to disenable this "NORMAL" source whenever the address is to be selected from the Direct Address Selection Unit.

Figure 31 shows the circuit to perform the required function. The address is selected with the help of 12, two-position switches connected at the inputs of 12 tri-state buffers. One position of each switch is connected to +5V and the other one is grounded. The "LOAD ADDRESS" switch, on the maintenance panel of the processor, presets a flip flop which disenables the tri-state outputs of the

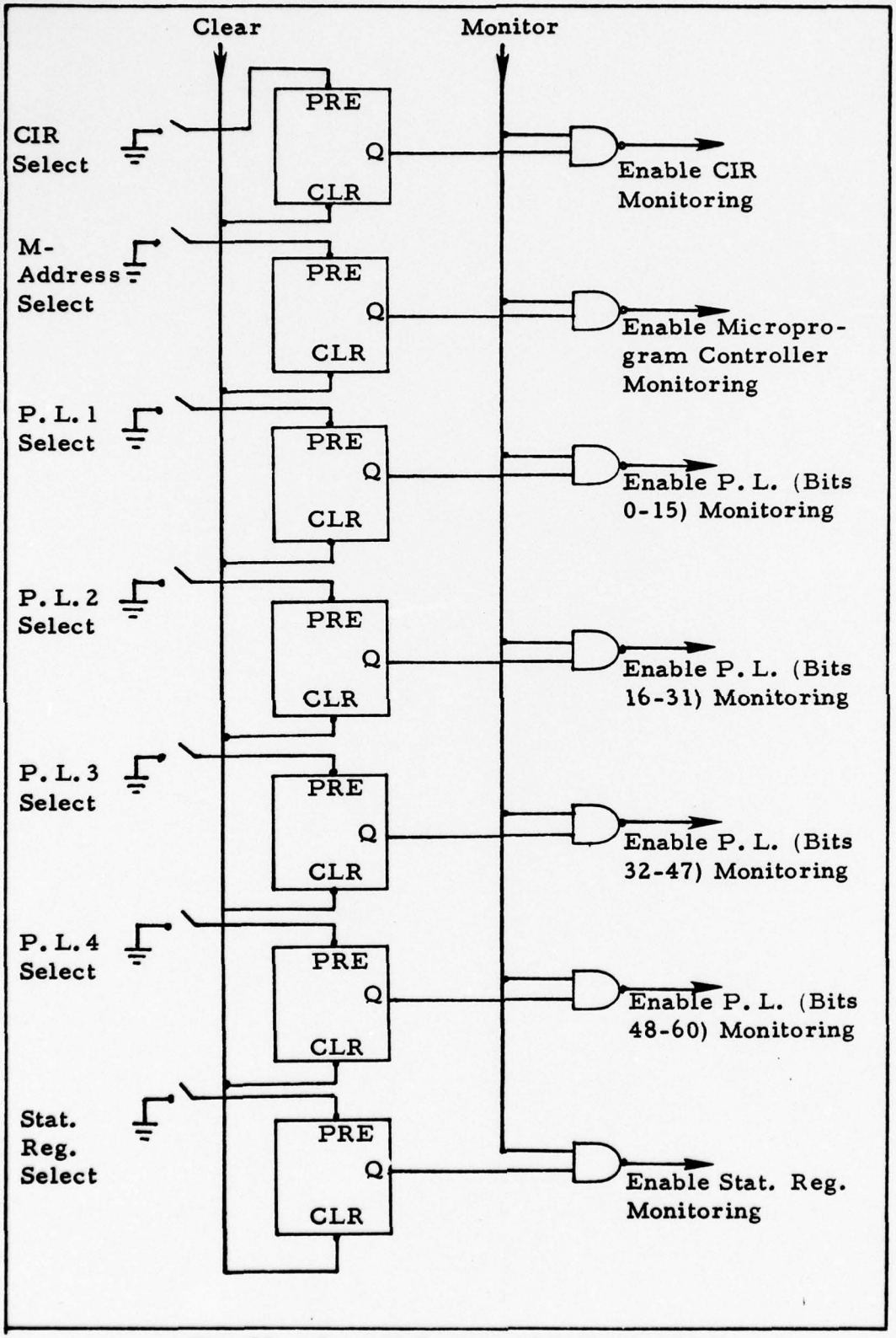


Fig. 30. Special Register Monitoring Unit

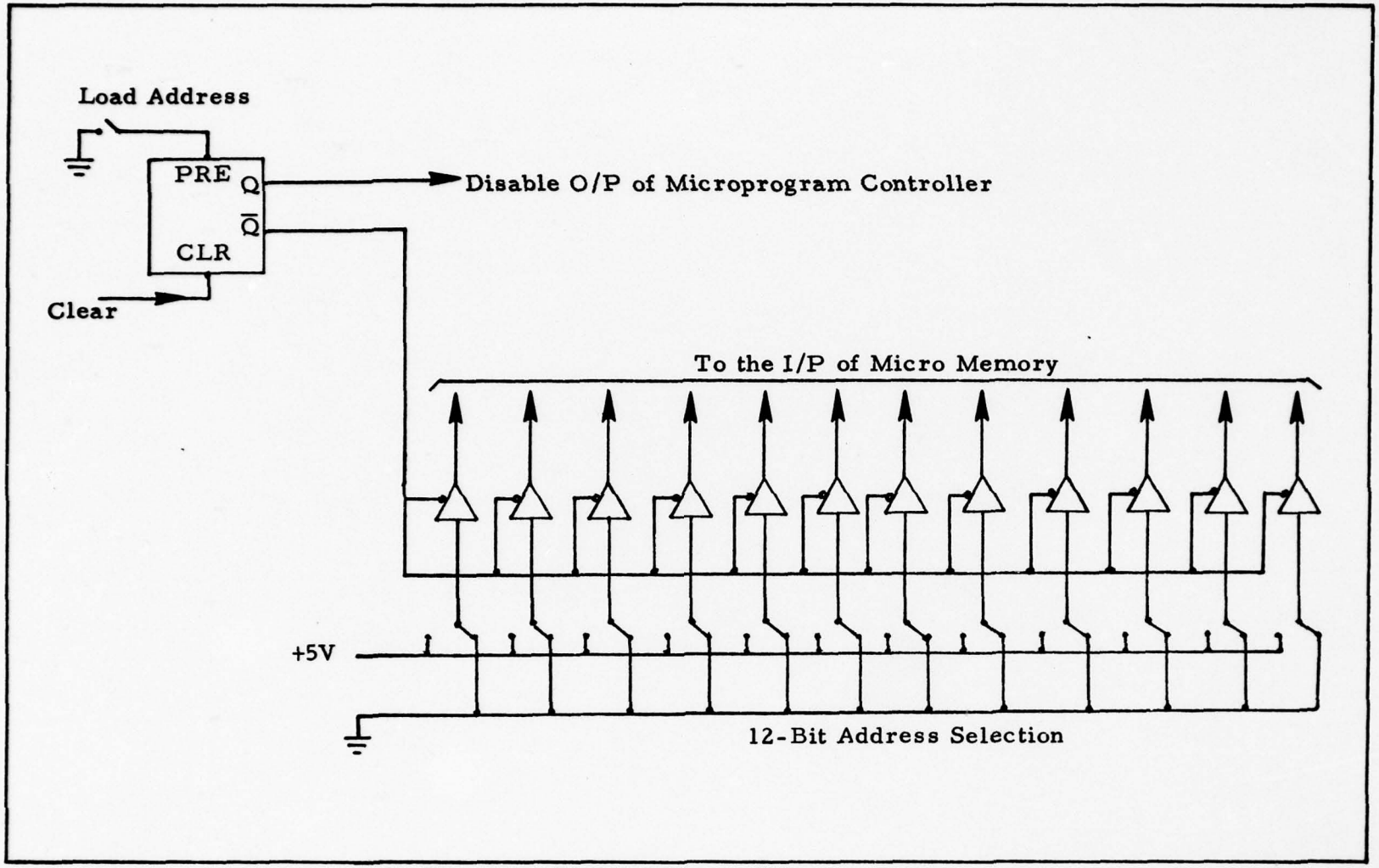


Fig. 31. Direct Address Selection Unit

Microprogram Controller and applies the selected address to the 4-K Microprogram Memory. By applying a clock pulse, the contents of the selected address can be loaded into the Pipe Line Register which may be monitored as explained earlier.

Manual Clock Generator. Whenever the processor is in the "MONITOR MODE," the Master Clock is inhibited from the ALU and the Microprogram Controller. Instead, a clock pulse is generated manually when desired and applied to the processor system.

Figure 32 shows the manual clock generator. When the processor is not in "Monitor Mode," the Master Clock is available to the system. If MONITOR switch is pressed, it is inhibited from the system but still clocks the flip flop M. The flip flop L may be preset by the switch "Manual Clock." Thus when L is preset, logic one is applied to the D input of M which appears at its Q output when the positive going edge of the Master Clock is applied. At the same time, the $\overline{Q}$ output of M goes LOW and resets the flip flop L. This places logic zero at the D input of M. On the positive edge of the next Master Clock pulse, the Q output of the flip flop M goes to logic zero. The net result is a positive output pulse on the Q output of flip flop M that lasts for one, and only one, whole Master Clock interval.

Using Manual clocking in the "Monitor Mode" gives the additional flexibility of continuously monitoring a particular output for a sequence of microinstructions by just pressing the Manual Clock

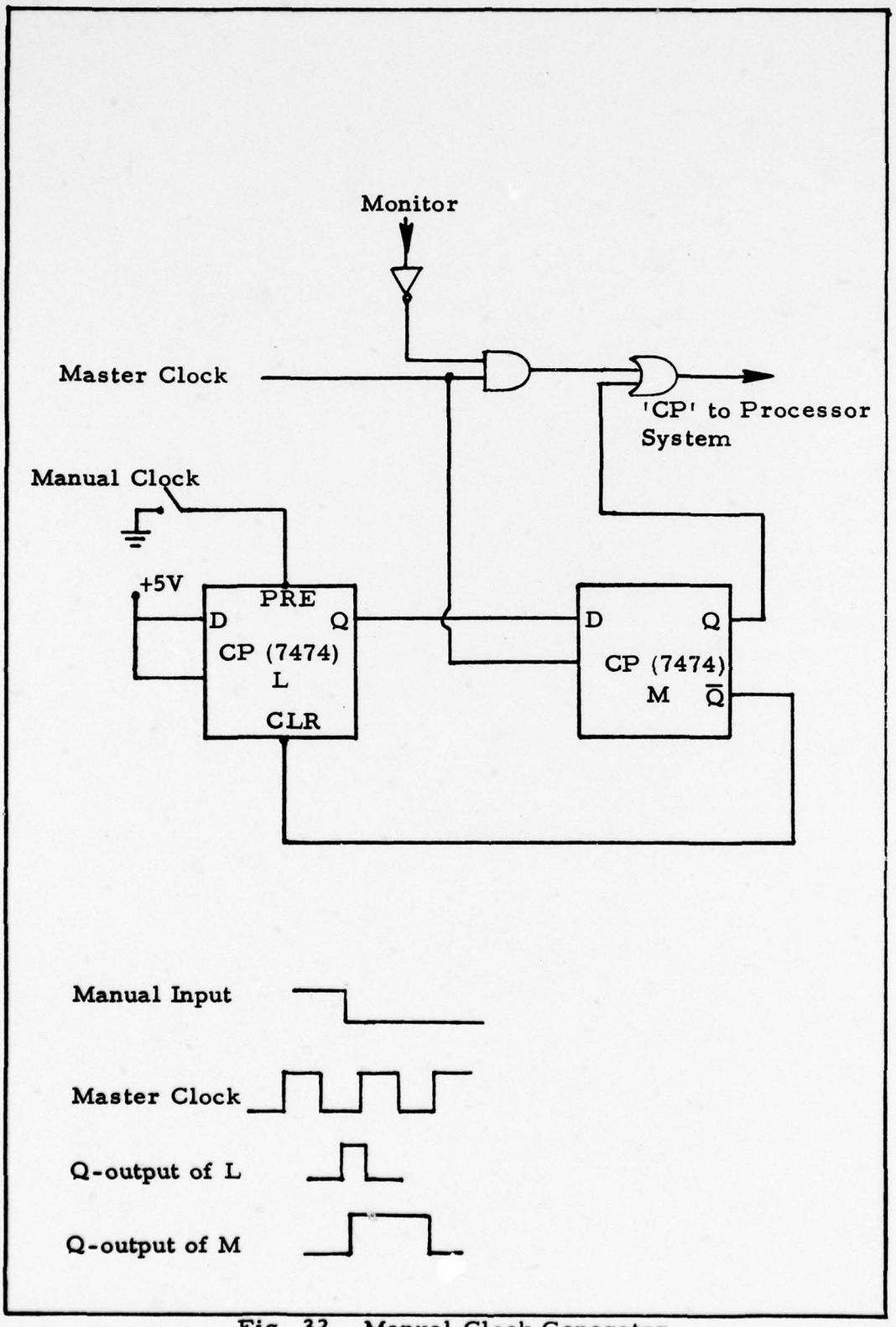


Fig. 32. Manual Clock Generator

switch for each microinstruction.

Summary

This chapter has attempted to outline the functions and the limitations of the Micro Level Monitor for the DP/M processor. A design has been presented to meet the same objectives.

The next chapter presents the conclusion of the present design of the processor and offers some recommendations for subsequent improvement.

VI. Conclusion and Recommendations

This chapter summarizes the design of the processor. It also gives the recommendations, based on the present design approach, to further improve the processor implementation using Am 2900 chip set.

Design Summary

This report covers the sequential development of the DP/M processor design. The design was realized using Am 2900 microprocessor chip set. The processor is a 16-bit, two's complement, fixed point, eight register file architecture. It provides a set of 41 macroinstructions to perform arithmetic, logical, shift, and data-transfer operations. The operands are derived either from the register file or from the Main Memory. The results may be placed into either the register file, Main Memory, or transferred as input/output data. The processor handles its own internal interrupts and also supports the external interrupts generated by the BIU and the input/output devices.

The processor design consists of four CPU slices (Am 2901), one carry look-ahead generator (Am 2902), one microprogram controller (Am 2910), and eight PROMS (Intel 3604A-2) to make up the microprogram memory. The microprogram memory stores the 64-bit microinstructions to control the fetching and execution of the

macroinstructions. The eight-register file was formed using the first eight words in the 16-word RAM of ALU. Other logic units like multiplexers, decoders, PROMs, and registers were used to augment the basic processor hardware. This was done to speed up the execution of certain instructions like Multiply, Divide, Shift, Branch-On-Condition, and all the Extended Short Format instructions requiring sign-extension of the I-Field. The processor hardware has been organized to implement first level pipeline mode of operation. This feature provides the microinstruction look-ahead capability to the Microprogram Controller.

For microprogramming purposes, the RTL flow charts were prepared first, and then micro-codes were written. The flow charts and the micro-codes have been arranged according to the various phases of the processor operation, namely the Power-up, Instruction-Fetch, Operand Derivation, Execution, and the Interrupt Handling.

The processor design also incorporates a Micro-Level Monitor. This facility could be used to display the contents of various registers while manually executing the microinstructions.

Conclusion

Based on this design study, it was found that the Am 2900 Microprocessor Chip Set provided a really powerful, flexible, and a compatible emulating source for the DP/M Processor. The two features of the ALU Chip (Am 2901), namely, the register to register

operation within the register file and the availability of the status information (sign, overflow and zero) at the end of ALU operation, were found to be very helpful. They helped to reduce the number of microinstructions. Similarly the Microprogram Controller Chip (Am 2910) alone provided the important functions of a micro-sequencer, next address selection logic, and a loop counter. The presence of a LIFO push/pop stack added further power to that chip. This feature allowed the efficient execution of the nexted micro sub-routine linkages. The use of Am 2910 greatly simplified the proces-sor design.

Adding up all the time-delays in the processor circuit, it was found that a 250-nanosecond clock would meet the requirements for fetching the next microinstruction into the pipeline register, and simultaneously executing the present one in the ALU. Now to meet the specified figure of executing 250 kilo instructions per second, each macroinstruction gets a maximum of 4 microseconds to com-plete. With 250 nanosecond clock, the processor is capable of exe-cuting 16 microinstructions in 4 microseconds. A review of the micro codes (Ref Appendix C) indicates that all the given macro-instructions, with the exception of MULTIPLY and DIVIDE, would need less than 16 microinstructions to complete. Hence it is inferred that a 250-nanosecond clock would be quite suitable for the processor. For MULTIPLY and DIVIDE instructions, an alterna-tive is suggested later in this chapter.

It is, therefore, concluded that the Am 2900 Microprocessor Chip Set is quite suitable and the design, presented in this report, can be hardwired to realize a Lab Model of the DP/M processor.

Recommendations

The present design conforms to the specifications laid out for the DP/M Processor. The following recommendations, however, are made to simplify the processor design and to further improve its performance.

Change in Interrupt Scheme. In the specified scheme, the processor handles its four internal interrupts. It maintains the Interrupt Mask for the internal as well as for the external interrupts. The mask bits meant for external interrupts are transferred to the respective devices. Thus, whenever an external device wants to interrupt the processor, it checks the corresponding mask bit and generates an interrupt request. It also provides a trap vector which the processor receives to start the external interrupt servicing routine. Since the trap vector is transmitted on the data lines which are a component of the I-Bus, it is essential that the interrupting device must first get control of the I-Bus. If the I-Bus is already being used by a high priority device like BIU, then the interrupting device may have to waste a lot of time before it can be serviced.

In order to avoid this, it is recommended that the processor be made responsible for generating the trap vectors for all external

interrupts as it does for its own internal interrupts. It should receive the external interrupt requests directly as its interrupt control unit (Am 2914). This feature will simplify the interrupt structure. The processor will not be required to load the Mask Registers of the external devices. The devices will not waste time to get access to the I-Bus for sending the trap vector.

The Am 2914 automatically prioritizes 8-interrupt requests and generates vector addresses. To implement the aforementioned change, another two or three Am 2914s are required to be cascaded with the one chip already used by the processor.

Using Am 2903 Microprocessor. The Am 2903 is the next generation bipolar microprocessor slice (expected to be commercialized by the middle of 1978). It performs all the functions of Am 2901 and, in addition, provides a number of significant enhancements that are especially useful in arithmetic-oriented processors (Ref 6:2). In addition to its complete arithmetic and logic instruction set, the Am 2903 provides a special set of instructions which facilitate the implementation of multiplication and division.

Using Multiply Special Functions, two N-bit, unsigned or two's complement numbers can be multiplied in N clock cycles. The multiplication uses conditional add and shift algorithm. No external hardware is required.

Similarly, the Divide Special Functions can be used to perform a two's complement, non-restoring divide operation. These

functions provide both single and double precision operations in N clock cycles, where N is the number of bits in the divisor. The correction of the quotient is also taken care of.

Bibliography

1. Air Force Systems Command. Distributed Processor/Memory Architectures Design Program. AFAL-TR-75-80. Wright-Patterson Air Force Base, Ohio: Air Force Avionics Laboratory, February 1975.

2. Advanced Micro Devices Incorporated. Am 2900 Bipolar Microprocessor Family. Sunnyvale, California: 1976.

3. Advanced Micro Devices Incorporated. Am 2910 Microprogram Controller. Preliminary Data. Sunnyvale, California: May 1977.

4. Advanced Micro Devices Incorporated. Am 2914 Interrupt Encoder. Sunnyvale, California: 1976.

5. Maud, Azhar. Emulation of the DP/M Processor on Intel 3000 Microprocessor Chip Set. Unpublished thesis. Wright-Patterson Air Force Base, Ohio: Air Force Institute of Technology, December 1976.

6. Vernon, Coleman, et al. The Next Generation Four-Bit Bipolar Microprocessor Slice--The Am 2903. Advanced Micro Devices, Sunnyvale, California: August 1977.

Appendix A

Processor Specifications

Definition of Terms

A	Second half of a 32 bit instruction which is used as an address, data, or displacement.
C	Command field used to specify the desired operation.
CAW	Command Address Word; used to specify a device address with the I/O instructions. CAW is an 8 bit field.
CCR	Condition Code Register; the two MSBs of the status word. The CCR contains the sign and zero indications of previous results and is tested by the conditional branch instructions.
DA	Derived Address; the address of the operand derived during address calculations.
DO	Derived Operand; the operand derived during address calculations. DO = (DA).
I	Immediate data; signed (+127 - 128) immediate data or displacement.
ISP	Interrupt Stack Pointer; the contents of register six (R6) is used as a push down stack pointer.
LSB	Least Significant Bit, usually the right most bit.
M	Mode select field for addressing.
MSB	Most Significant Bit; usually left most bit.
PC	Program Counter; general register seven (R7) is the program counter.
R	R field of instruction; used to designate one of the eight general purpose registers.

S — Refers to 'sign' bit of CCR.

SW — Status Word; the status word is selectively modified during program execution as described below. It is also set by any interrupt or the execution of an exchange status and PC or a return from interrupt instruction. Further, the CAW of 00 (HEX) is reserved for assessing the portion of the status word external to the processor. The status word has the following format:

			6			10	Bits/Field
S	Z	OF	OIM	DEM	MEM	IM	

S:Z — On those instructions that change the CCR, S is set equal to the sign (MSB) out of the ALU, and Z is set to a one if the output of the ALU is all zero's and zero otherwise.

OF — Overflow Flag which is set by any arithmetic overflow and is cleared by any arithmetic instruction that does not create an overflow or by any conditional branch instruction that tests the overflow condition. The overflow flag is also set or cleared by an exchange status and program counter instruction or by a return from interrupt instruction.

OIM — Overflow Interrupt Mask which is set or cleared by an exchange status and program counter instruction or by a return from interrupt instruction. If an arithmetic overflow occurs and OIM = 1 and the next instruction does not test the state of the overflow flag, then an overflow interrupt will occur.

DEM — Device Error Mask which is set or cleared by an exchange status and program counter instruction or by a return from interrupt instruction. If an addressed device on the PE internal bus does not respond with an acknowledge in a specified time limit and the DEM = 1, the device error interrupt would occur.

MEM — Memory Error Mask which is set or cleared by an exchange status and program counter instruction or by a return from interrupt instruction. If a memory error occurs and MEM = 1, the memory error interrupt would occur. A memory error could result if (1) an addressed memory location does not respond with an acknowledge in

a specified time limit, or (2) a memory parity error is detected, or (3) a memory write protect error is detected.

IM — Ten Bit Interrupt Mask. If multiple interrupts are enabled and occur simultaneously, then invalid command overflow interrupts take precedence over other interrupts, and the interrupt that corresponds to the MSB(S) of the interrupt mask take precedence over those that correspond to the LSB(S). A one in an IM bit position shall enable the corresponding interrupt, a zero shall disable it. The IM is set or cleared by an exchange status and program instruction, a return from interrupt instruction, or an output instruction with a CAW of 00 (HEX).

T — T field of instruction; a three bit field used to designate one of the eight general purpose registers.

X — A two bit field used to specify one of four instruction formats.

Z — Refers to 'zero' bit of CCR.

(Arithmetic operations are two's complement 16 bit, unless specified.)

ADD

OPERATION: (R) ← (R) + DO

CCR CHANGES:

S:Z	If Result
00	.GT. 0
01	.EQ. 0
10	.LT. 0

OVF: Set if sign of result differs from carry out.

AND

OPERATION: (R) ← (R) .AND. DO

CCR CHANGES:

S:Z	If Result
00	.GT. 0
01	.EQ. 0
10	.LT. 0

OVF: No change.

BRANCH CONDITIONAL

OPERATION: PC ← DA

WHEN	AND R =
CCR = 00	000
CCR = 01	001
CCR = 10	010
Overflow	011
CCR ≠ 00	100
CCR = 01	110
No Overflow	111

OVF: No change.

CCR CHANGE: No change.

BRANCH INDIRECT AND
LINK TO SUBROUTINE

OPERATION: (R)←PC; PC ←(I)

CCR CHANGES: Unchanged

Note: I is used as an unsigned number (0-255)

OVF: Unchanged

COMPARE SIGNED

OPERATION: (R) - DO (No destination)

CCR CHANGES:

S:Z	IF
00	(R).GT.DO
01	(R).EQ.DO
10	(R).LT.DO

OVF: No change

CLEAR BIT LOWER BYTE

OPERATION: (DA)←(DA).AND.(.NOT.2**(7-R))

CCR CHANGES:

S:Z	IF
00	Bit was previously zero
01	Bit was previously one

OVF: No change

CLEAR BIT UPPER BYTE

OPERATION: (DA)←(DA).AND.(.NOT.2**(15-R))

CCR CHANGES:

S:Z	IF
00	Bit was previously zero
01	Bit was previously one

OVF: No change

DIVIDE

OPERATION: Quotient (R+1), Remainder (R)←(R),(R+1)/DO

CCR CHANGES:

S:Z	When Quotient
00	.GT.0
01	.EQ.0
10	.LT.0

Note: (R),(R-1) form a double signed 32 bit integer. (R) = MSH, (R+1) = LSH. Sign of remainder will agree with original dividend.

OVF: Quotient carry out from bit 32 different than sign. CCR not modified when OVF occurs.

EXCHANGE SW AND PC

OPERATION: ISP←ISP-1; (ISP)←PC; PC←(DO) (first)
ISP←ISP-1; (ISP)←SW; SW←(DO+1) (second)

CCR & OVF are loaded as part of the new status word.

Note: The processor responds to an interrupt by executing an Exchange SW and PC with a DO from the input data lines.

OVF:

EXCLUSIVE OR

OPERATION: (R)← (R).XOR.DO

CCR CHANGES:

S:Z	When Result
00	.GT.0
01	.EQ.0
10	.LT.0

OVF: No change.

INCREMENT AND BRANCH
IF NEGATIVE SHORT

OPERATION: (R)← (R)+1; if (R). LT. 0 then PC← PC+I (sign extended) or else PC← PC+1

CCR CHANGES: No change

OVF: No changes

LOAD

OPERATION: (R)← DO

CCR CHANGES:

S:Z	When Result
00	.GT.0
01	.EQ.0
10	.LT.0

OVF: No change

LOAD ONE'S COMPLEMENT

OPERATION: (R)← One's complement of DO

CCR CHANGES:

S:Z	When Result
00	.GT.0
01	.EQ.0
10	.LT.0

OVF: No change

LOAD TWO'S COMPLEMENT

OPERATION: (R)← Two's complement of DO

CCR CHANGES:

S:Z	When Result
00	.GT.0
01	.EQ.0
10	.LT.0

OVF: No change

MULTIPLY

OPERATION: (R), (R+1)← (R+1)*DO

Note: (R) = signed MSH; (R+1) = signed LSH of result.

CCR CHANGES:

S:Z	When result (Sign Bit checked in (R))
00	.GT. 0
01	.EQ. 0
10	.LT. 0

OVF: Set when 8000 (HEX)*8000 (HEX) attempted, result is 8000 HEX in both registers.

MEMORY INPUT COMMAND

OPERATION: I/O control←CAW, (DA)←input

CCR CHANGES:

S:Z	When
00	Acknowledge received
01	No acknowledge received

Note: An external interrupt will occur if the external device does not respond in the allowed time and the I/O interrupt mask is enabled.

OVF: No change

MEMORY OUTPUT COMMAND

OPERATION: Output←(DA);I/O control←CAW

CCR CHANGES:

S:Z	When
00	Acknowledge received
01	No acknowledge received

Note: An external interrupt will occur if the external device does not respond in the allowed time and the I/O interrupt mask is enabled.

OVF: No change

MOVE & AUTOINCREMENT

OPERATION: ((R))←DO; (R)←(R)+1

CCR CHANGES: No change

OVF: No change

OR

OPERATION: (R)←(R).OR.DO

CCR CHANGES:

S:Z	When Result
00	.GT.0
01	.EQ.0
10	.LT.0

OVF: No change

POP MULTIPLE

OPERATION: (R)←(ISP), ISP←ISP+1
(R+1)←(ISP), ISP←ISP+1
⋮
} for T+1 number of registers.

CCR CHANGES: No change

OVF: No change

PUSH

OPERATION: (R)←(R)-1; ((R))←DO

CCR CHANGES: No change

OVF: No change

PUSH MULTIPLE

OPERATION: ISP ← ISP-1, (ISP) ← (R)
ISP ← ISP-1, (ISP) ← (R-1) for T+1 number of registers
'
'
'
'

CCR CHANGES: No change

OVF: No change

REGISTER INPUT COMMAND

OPERATION: I/O control ← DO; (R) ← input

CCR CHANGES:

S:Z	When
00	Acknowledge received
01	No acknowledge received

Note: DO contains the CAW information.

Note: An external interrupt occurs if the external device does not respond in the allowed time and the I/O interrupt mask is enabled.

OVF: No change

REGISTER OUTPUT COMMAND

OPERATION: I/O control ← DO; output ← (R)

CCR CHANGES:

S:Z	When
00	Acknowledge received
01	No acknowledge received

OVF: No change

RETURN FROM INTERRUPT

OPERATION: SW←(ISP); ISP←ISP+1
PC←(ISP); ISP←ISP+1

CCR CHANGES: Loaded as part of new status word.

OVF: Loaded as part of new status word.

Note: R6 is the ISP.

SUBTRACT

OPERATION: (R)←(R)-DO

CCR CHANGES:

S:Z	When Result
00	.GT. 0
01	.EQ. 0
10	.LT. 0

OVF: Set when result will not fit into a 16 bit number (sign differs from carry out).

SET BIT LOWER BYTE

OPERATION: (DA)←(DA).OR. 2**(7-R)

CCR CHANGES:

S:Z	When
00	Bit specified was previously 0
01	Bit specified was previously 1

OVF: No change

SET BIT UPPER BYTE

OPERATION: (DA)←(DA).OR. 2**(15-R)

CCR CHANGES:

S:Z	When
00	Bit specified was previously 0
01	Bit specified was previously 1

OVF: No change

SHIFT SINGLE

Shift

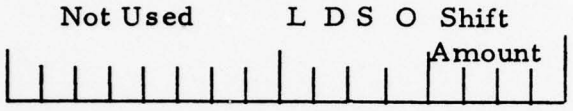
OPERATION: (R) ← (R), Shift/Rotate - control ←DO

CCR CHANGES:

S:Z	When Result
00	.GT. 0
01	.EQ. 0
10	.LT. 0

Note: The register specified is shifted/rotated as specified by the eight LSBs of the derived operand.

DO =



L Logical/Arithmetic 0 ⇒ Arithmetic 1 ⇒ Logical

D Direction 0 ⇒ Right 1 ⇒ Left

S Shift/Rotate 0 ⇒ Shift 1 ⇒ Rotate

Count Shift/Rotate Amount (0 to 15)

Arithmetic Rotates are not included.

OVF: Set when an arithmetic left shift out differs from the sign.

STORE

OPERATION: (DA) ←(R)

CCR CHANGES: No changes

OVF: No changes

STORE THROUGH MASK

OPERATION: (DA) ←((DA). AND. ($\overline{R}$)). OR. ((R+1). AND. (R))
where (R) = MASK and (R+1) = Data

CCR CHANGES: No changes

OVF: No changes

TEST BIT UPPER BYTE

OPERATION: Z←(DA).AND.2**(15-R)

CCR CHANGES:

S:Z	When
00	Designated bit is zero
01	Designated bit is one

OVF: No change.

TEST BIT LOWER BYTE

OPERATION: Z←(DA).AND.2**(7-R)

CCR CHANGES:

S:Z	When
00	Designated bit is zero
01	Designated bit is one

OVF: No change

SET STATUS WORD

OPERATION: SW←DO

CCR CHANGES: Loaded as part of the new status word.

Note: Interrupt testing is not done for one instruction following a set status command.

OVF: Loaded as part of the new status word.

READ STATUS WORD

OPERATION: (R)←SW

CCR CHANGES: No change

OVF: No change

Appendix B

Flow Charts

This Appendix contains the Flow Charts for the specified instructions (Ref Appendix A). The flow charts are arranged in the following sequence:

a. "Power Up" and "Fetch" phases

b. Operand Derivation phase

c. Execution phase

d. Interrupt Handling phase.

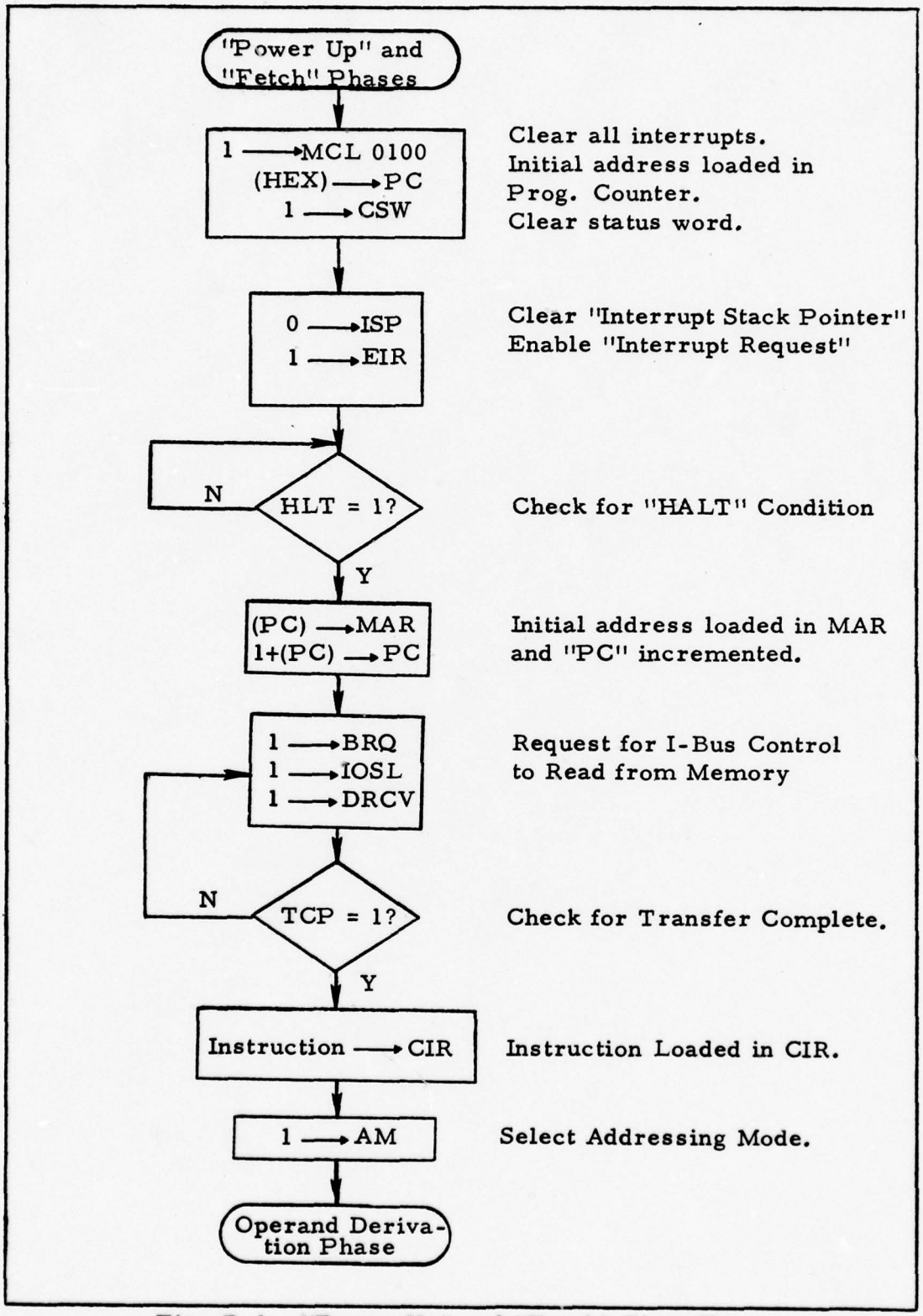


Fig. B-1. "Power Up" and "Fetch" Flow Chart

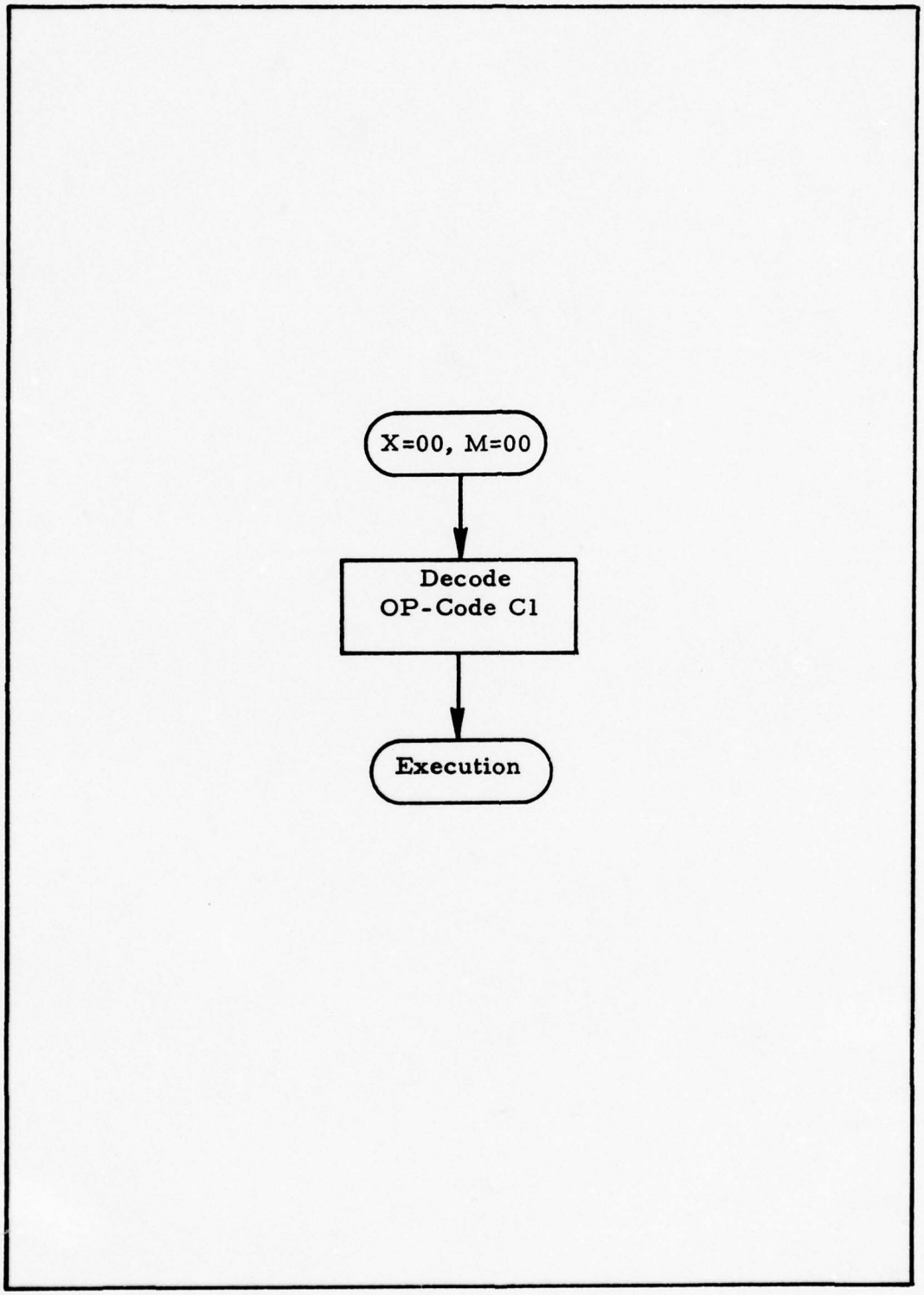


Fig. B-2. Operand Derivation "Register to Register" Mode

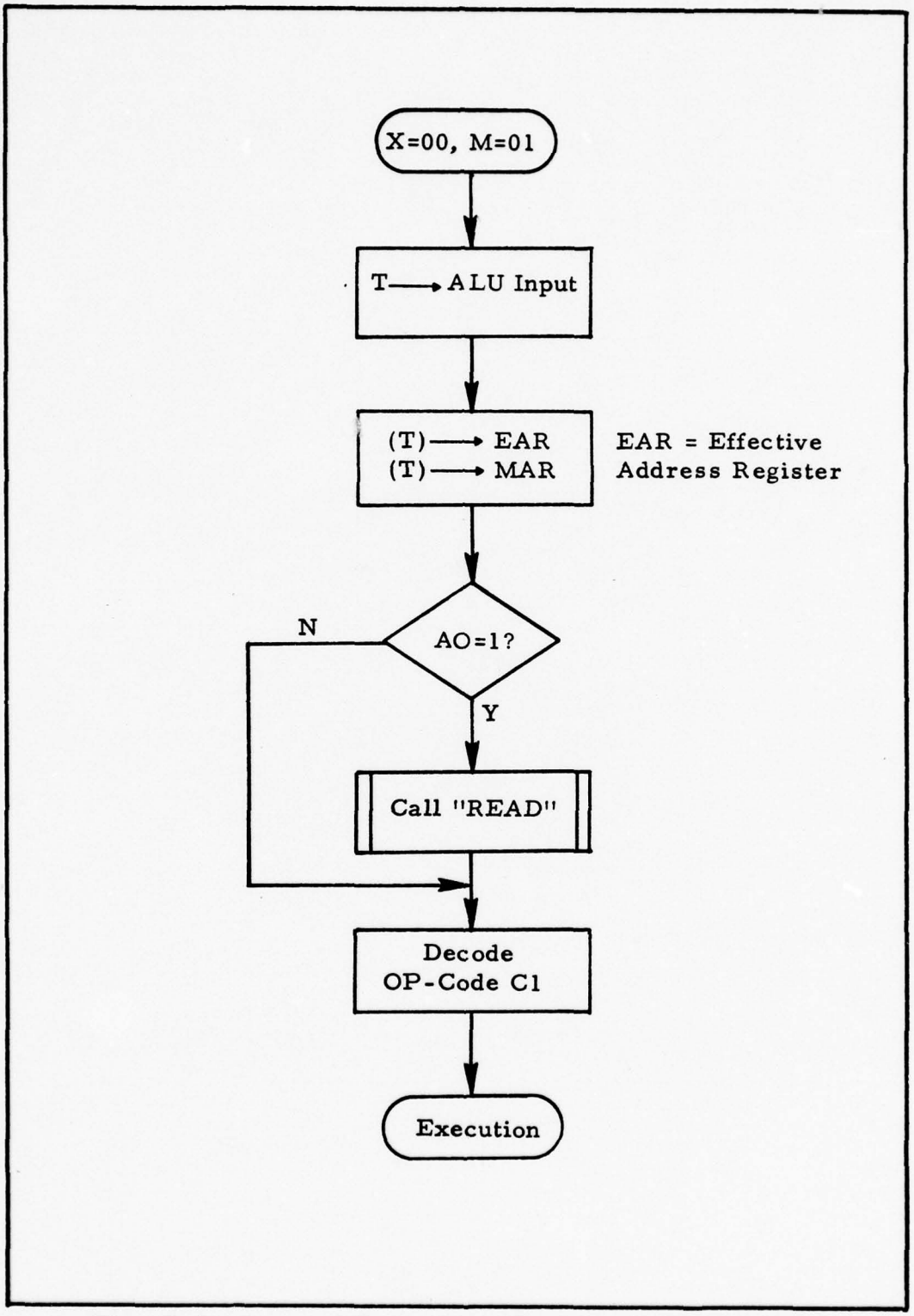


Fig. B-3. Operand Derivation "Register Indirect Mode"

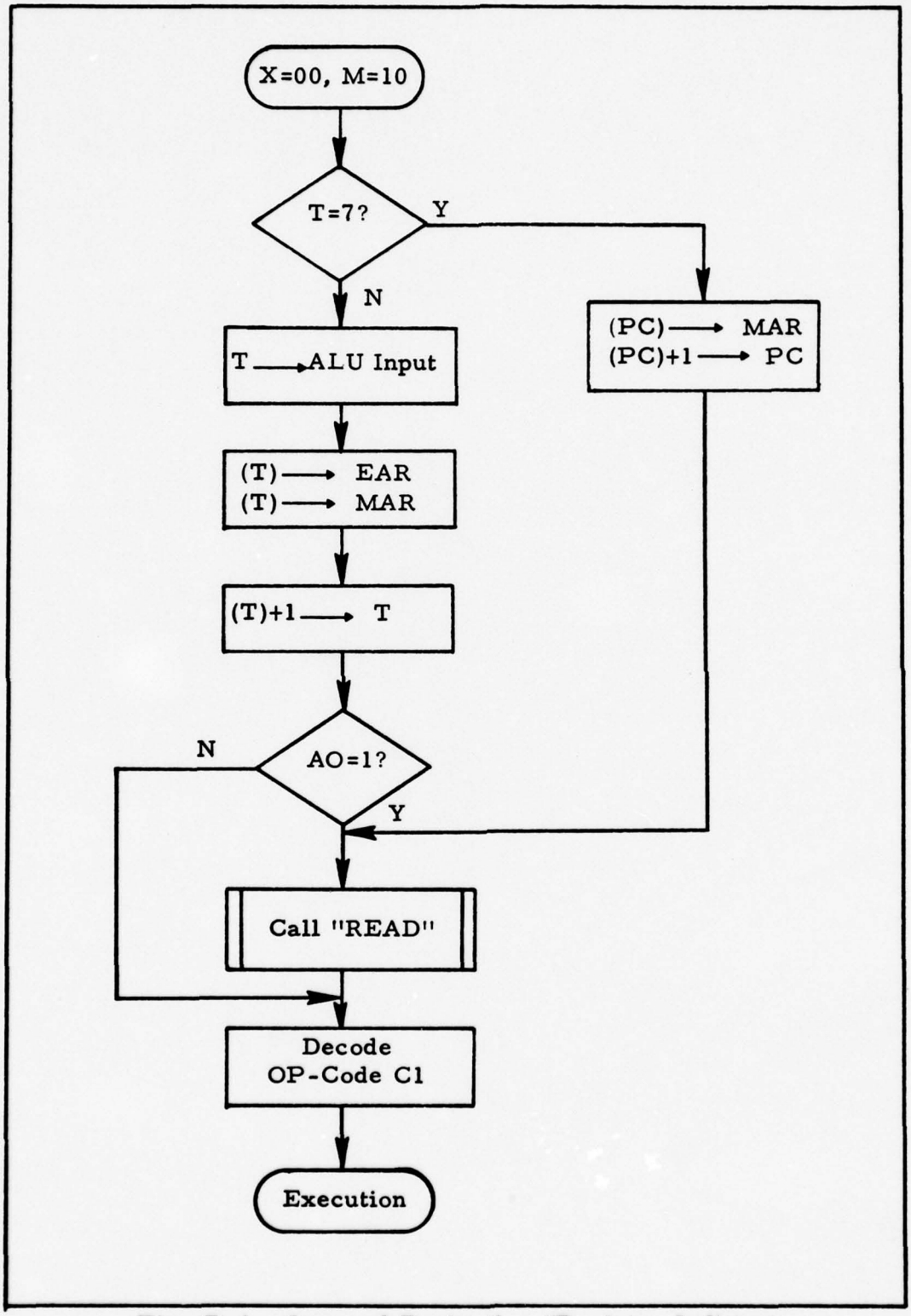


Fig. B-4. Operand Derivation "Register Indirect Autoincrement Mode"

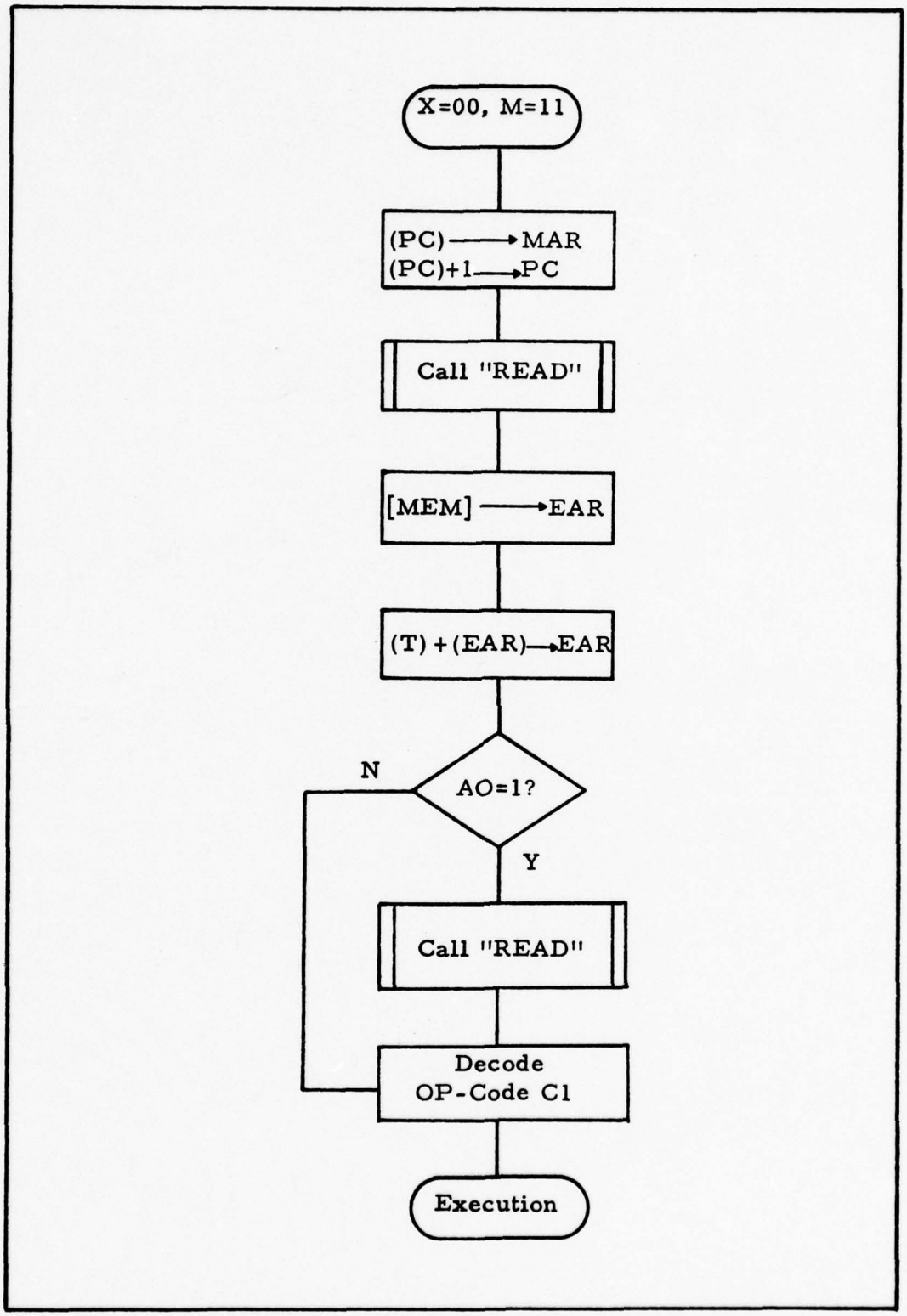


Fig. B-5. Operand Derivation "Direct and Direct Indexed Mode"

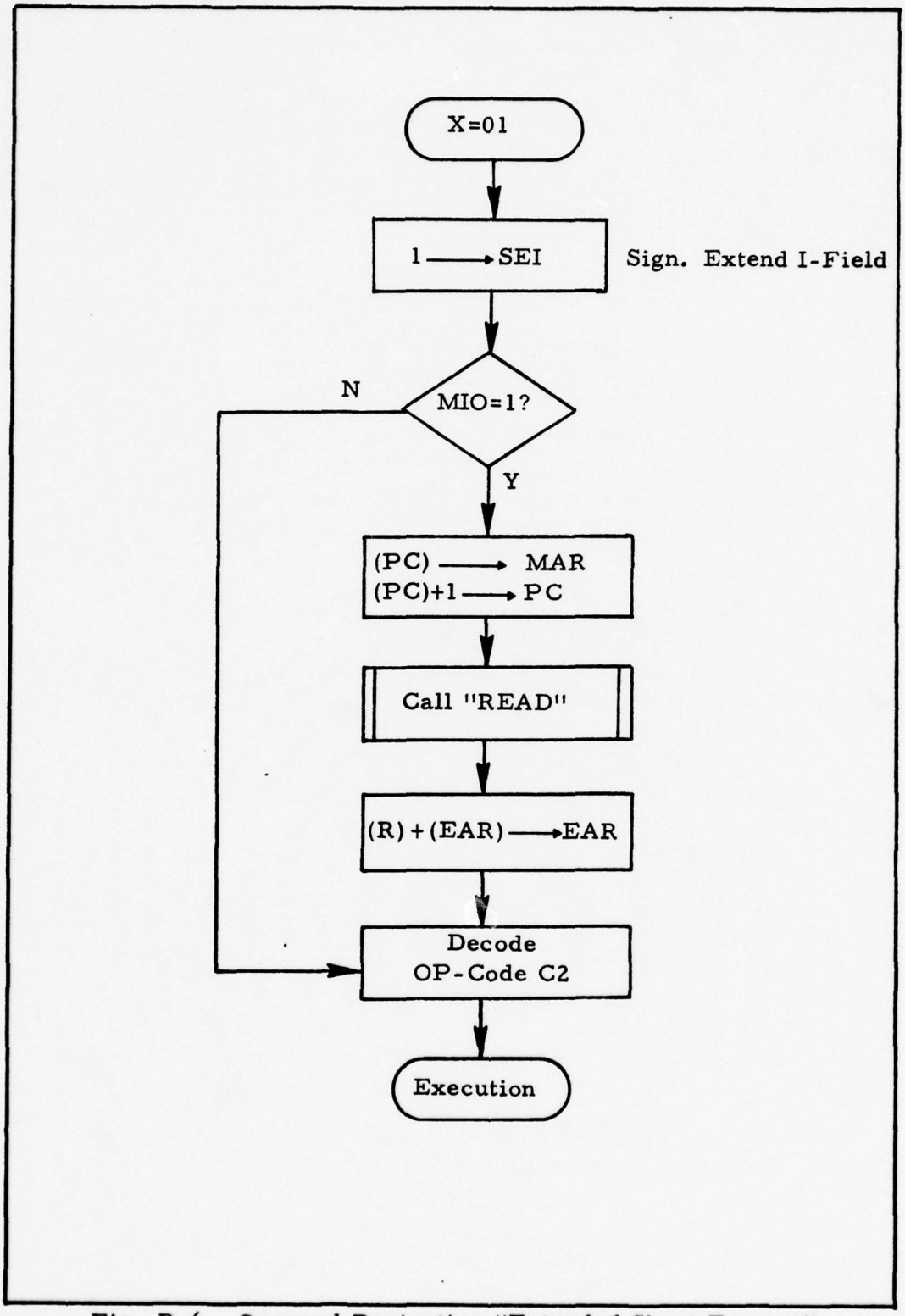


Fig. B-6. Operand Derivation "Extended Short Format"

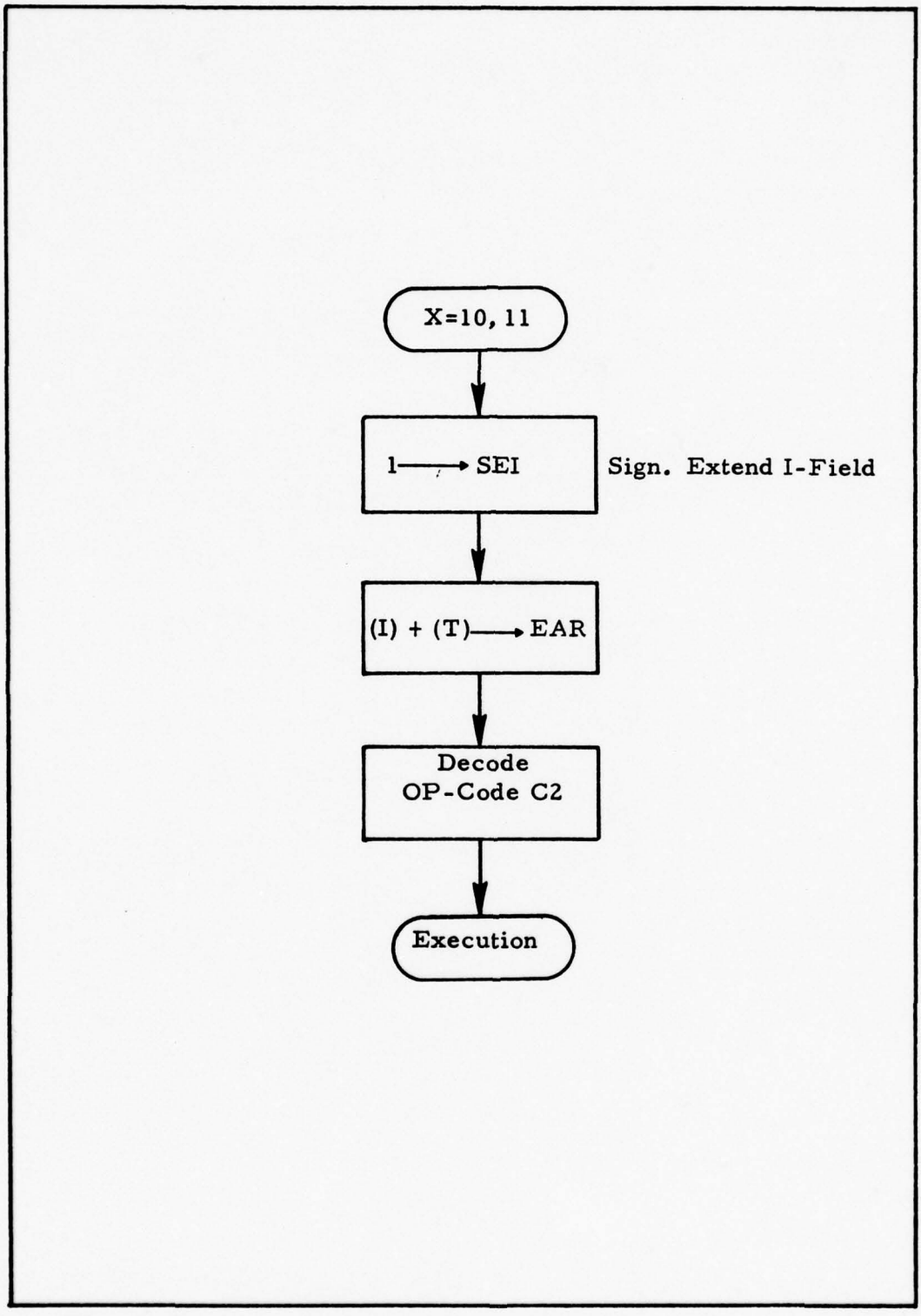


Fig. B-7. Operand Derivation "Load and Store Direct Mode"

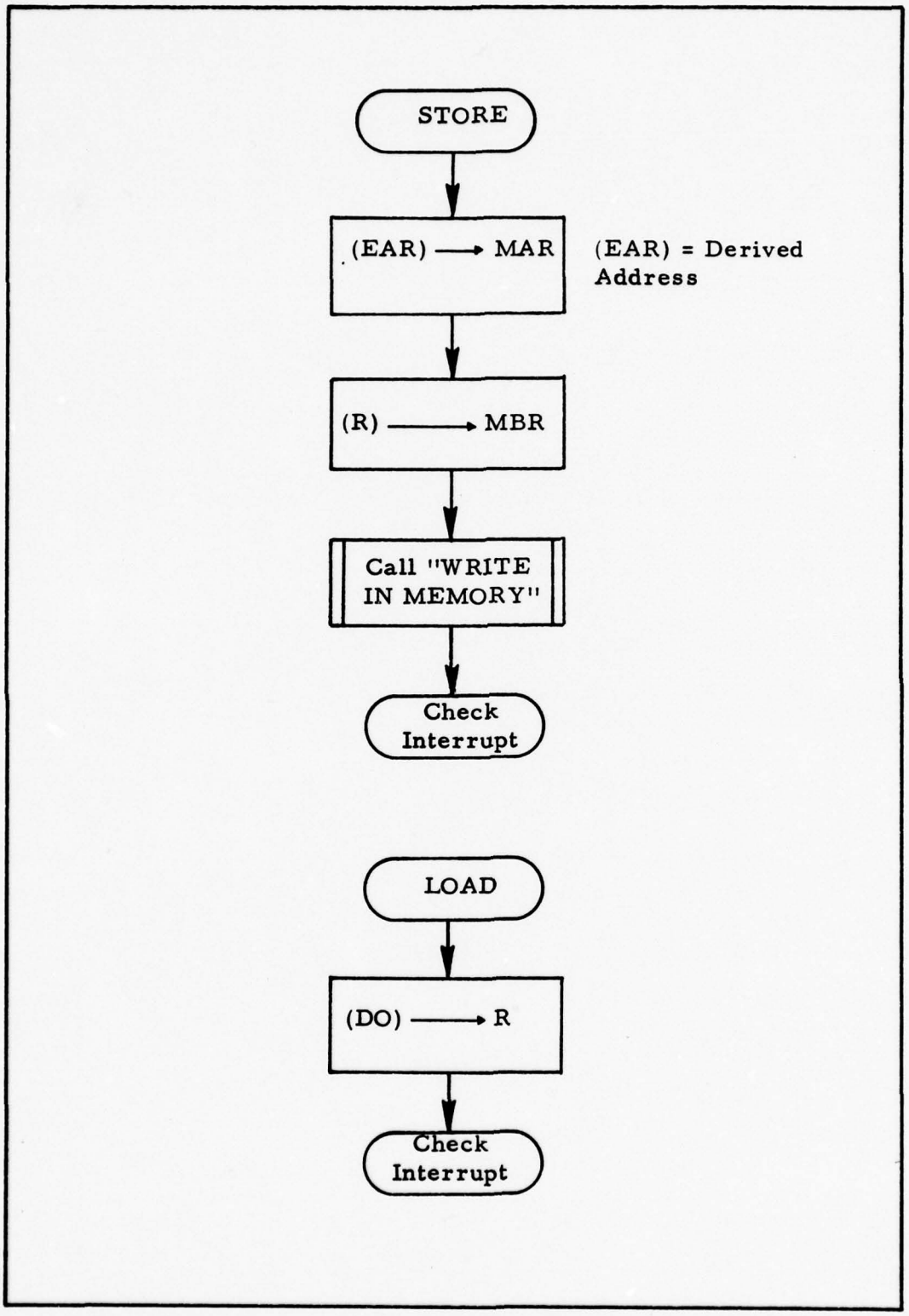


Fig. B-8. Execution Phase--Data Transfer Instructions

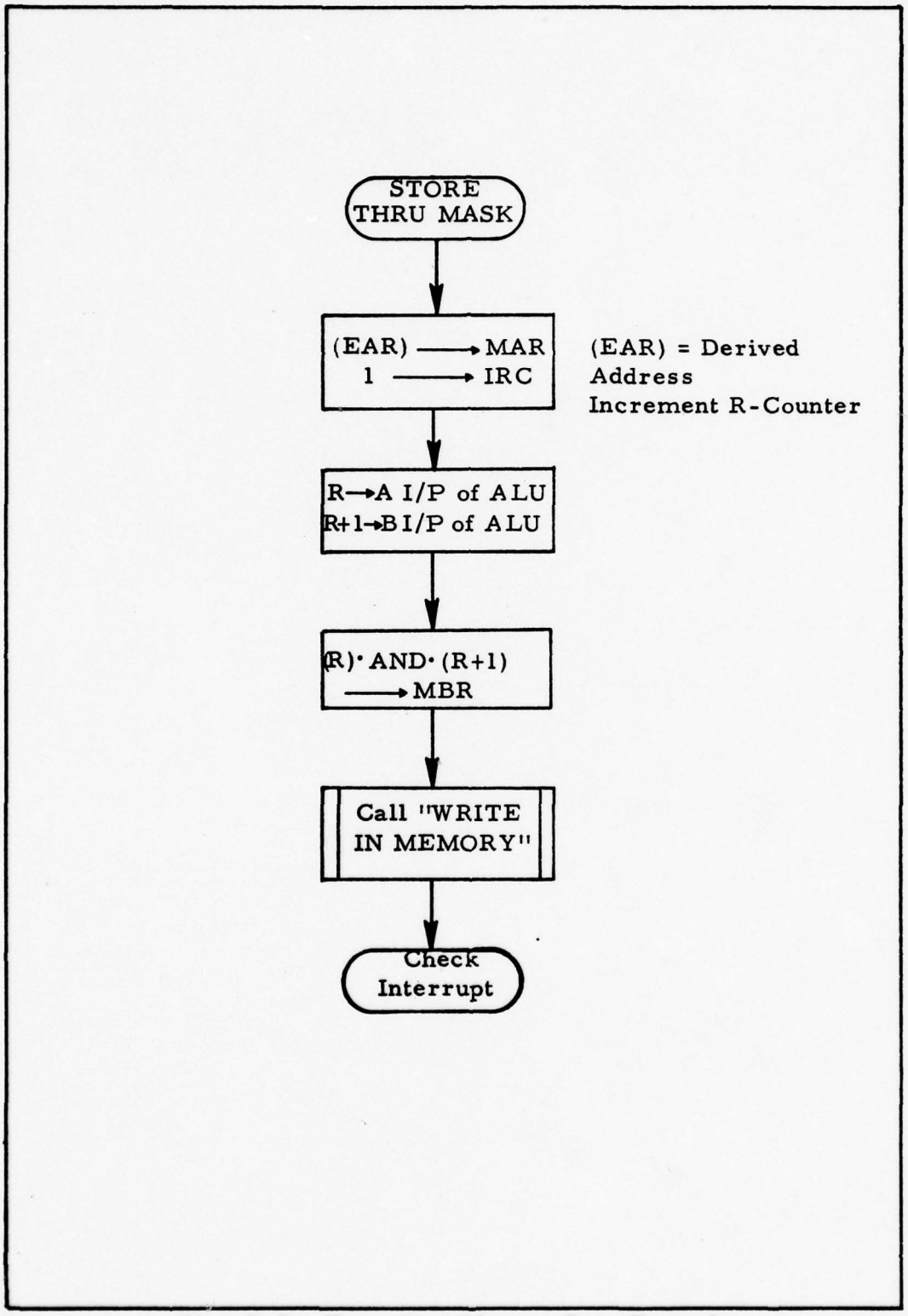


Fig. B-8. Execution Phase--Data Transfer Instructions (Continued)

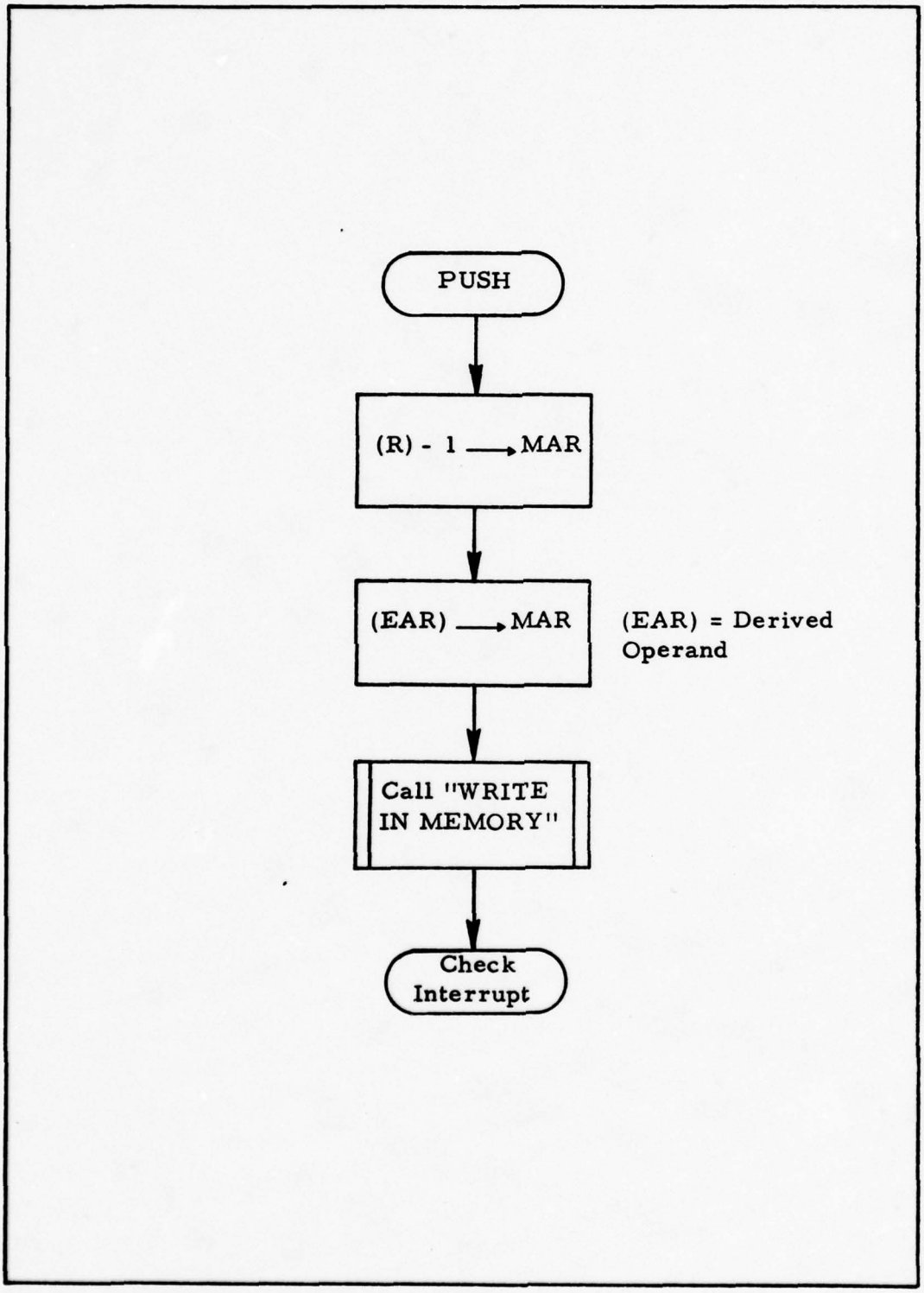


Fig. B-8. Execution Phase--Data Transfer Instructions (Continued)

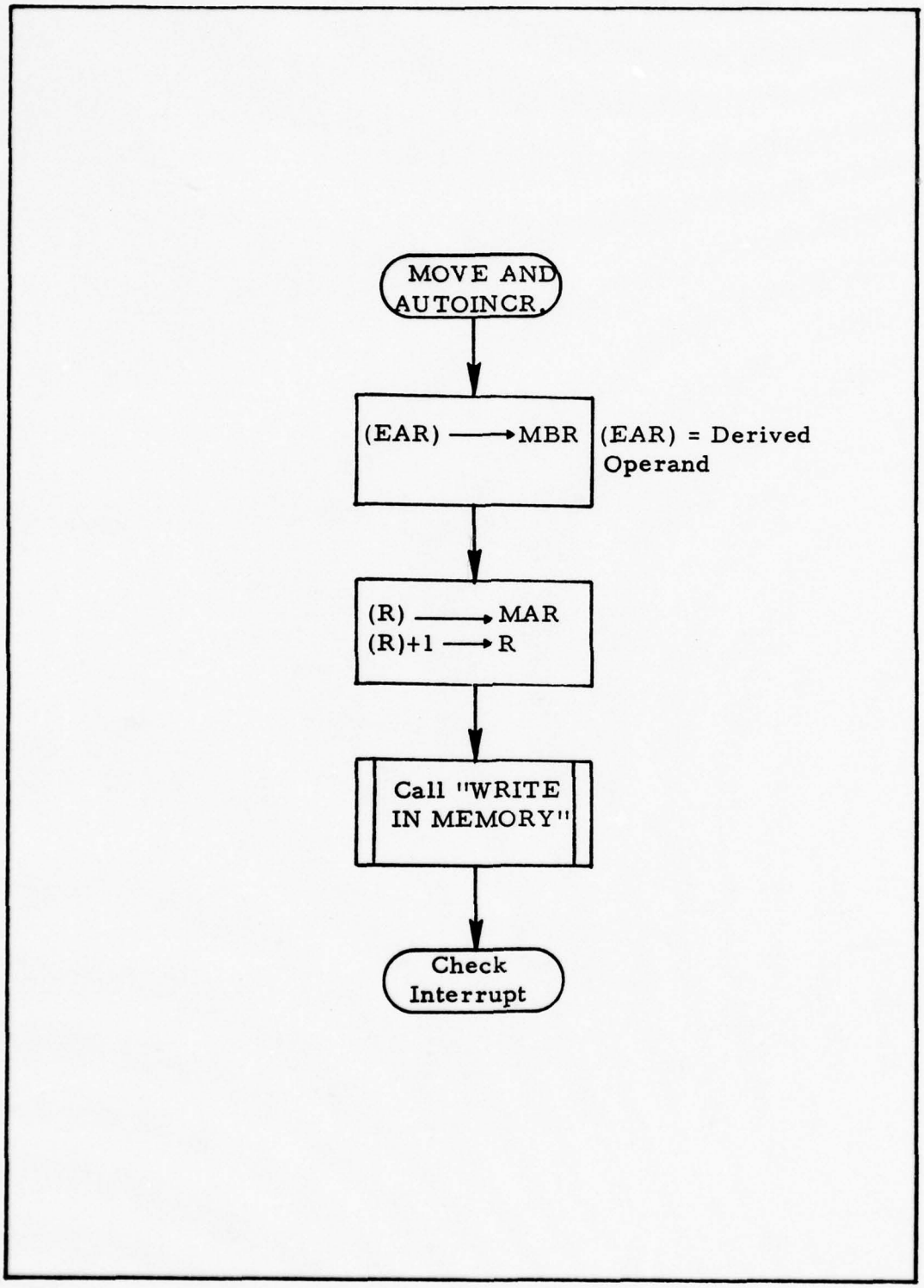


Fig. B-8. Execution Phase--Data Transfer Instructions (Continued)

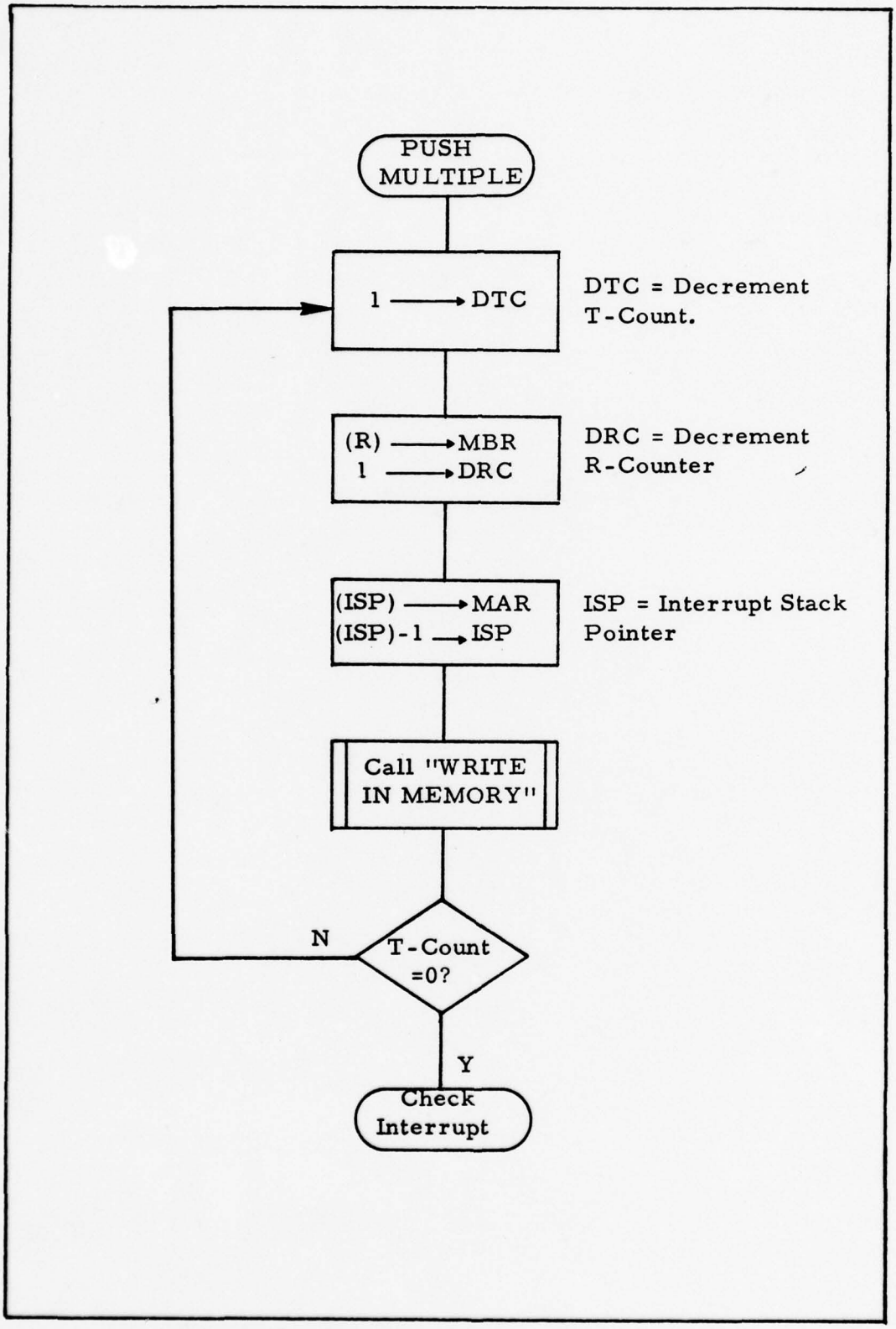


Fig. B-8. Execution Phase--Data Transfer Instructions (Continued)

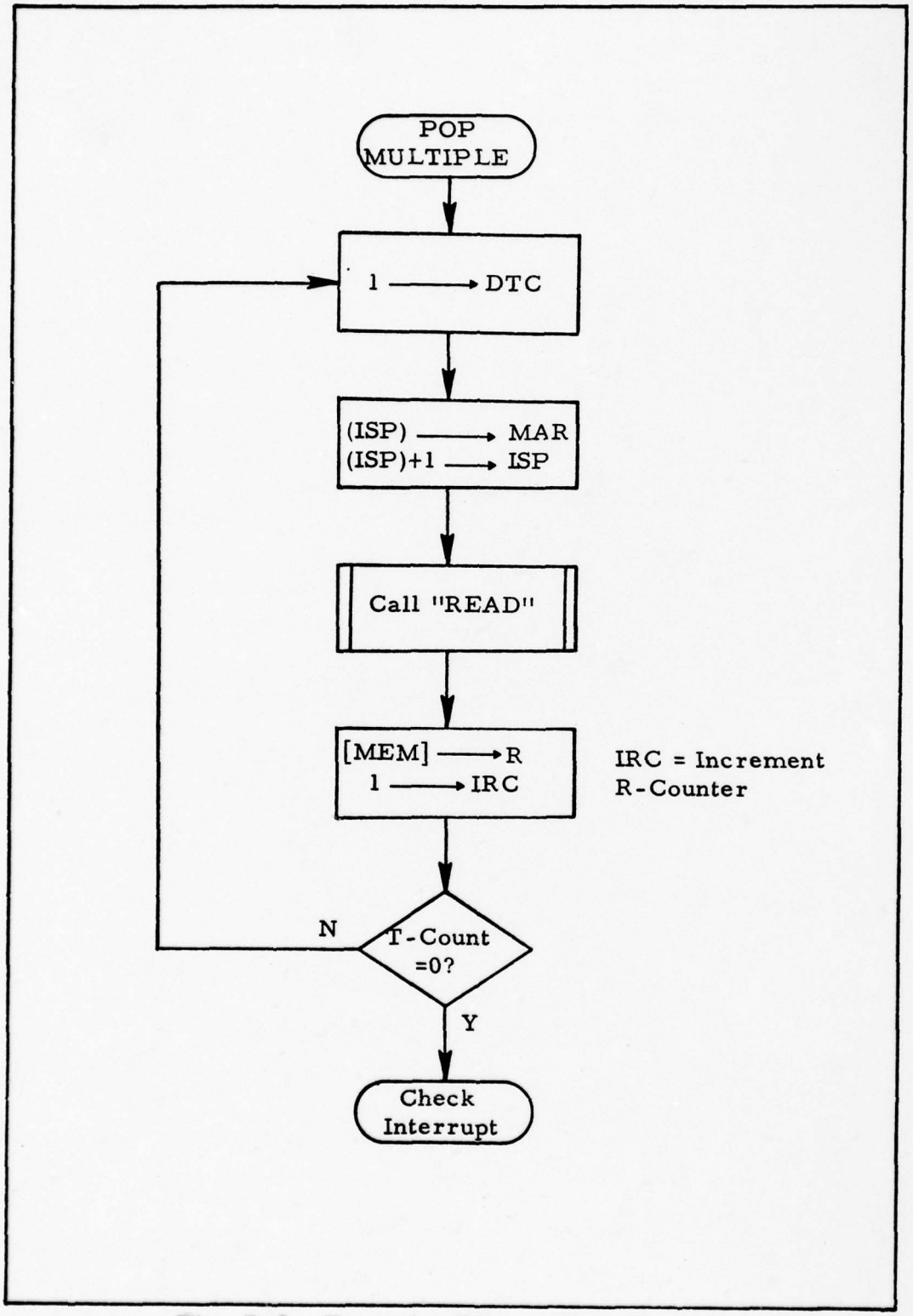


Fig. B-8. Execution Phase--Data Transfer Instructions (Continued)

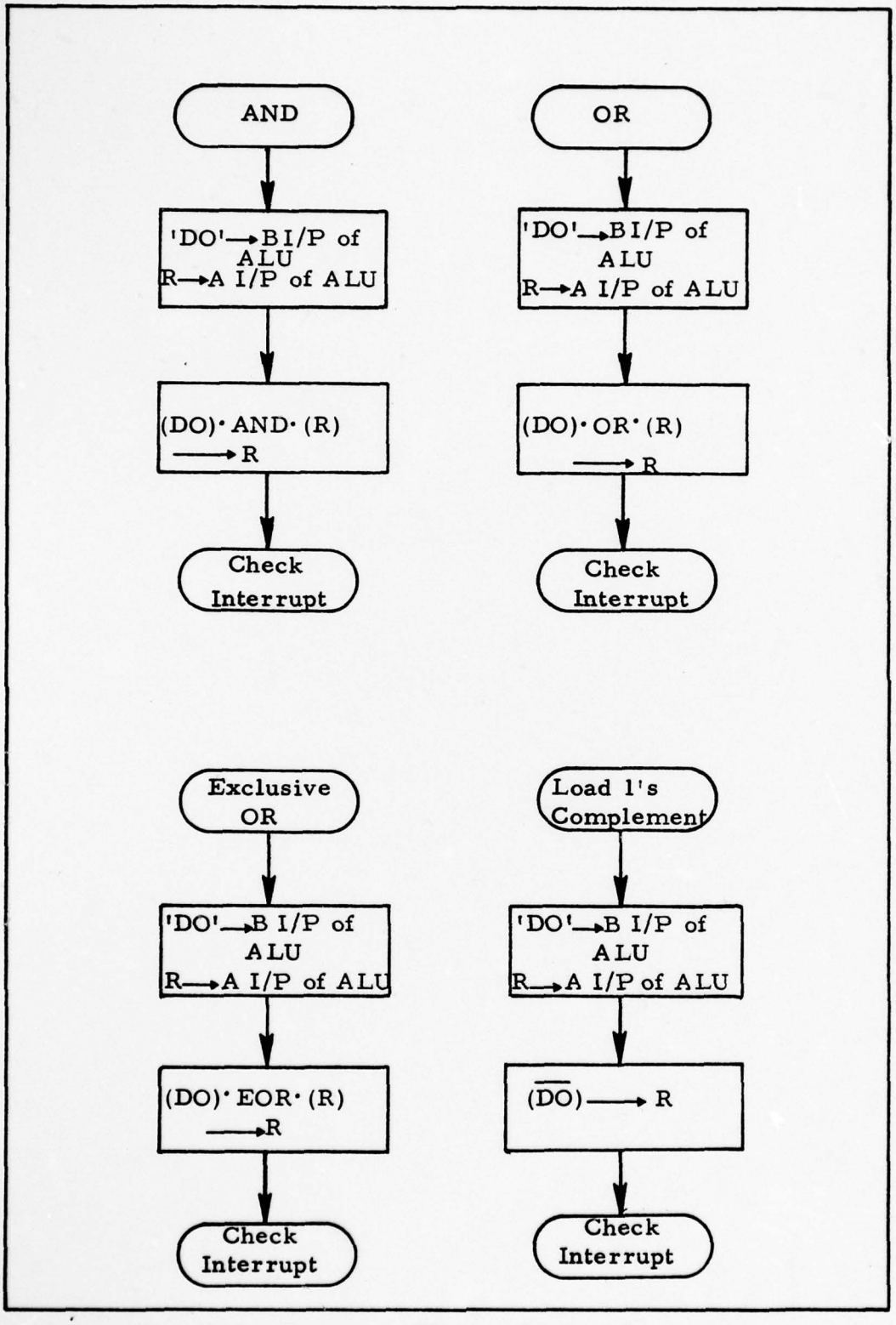


Fig. B-9. Execution Phase--Logical Instructions

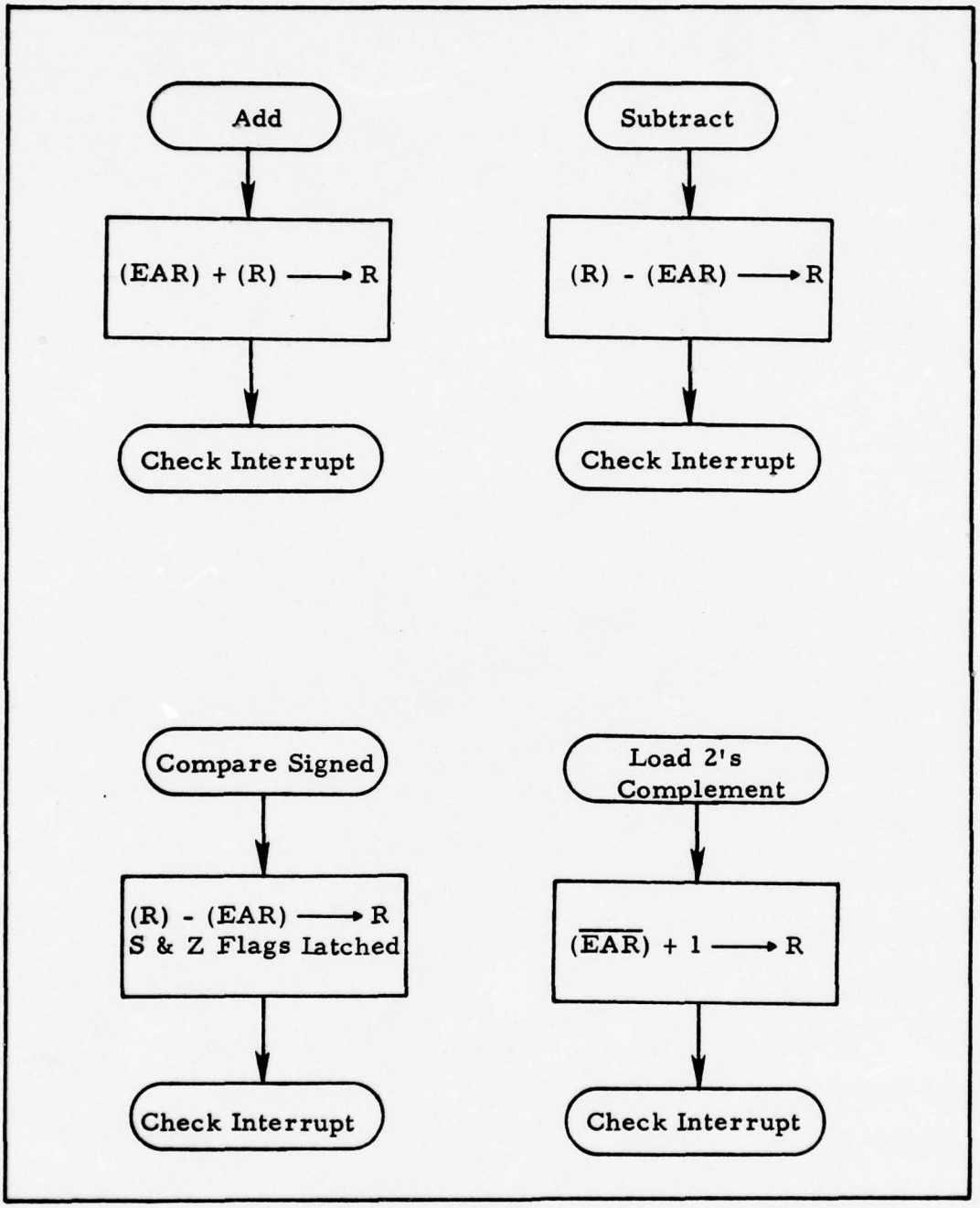


Fig. B-10. Execution Phase--Arithmetic Instructions

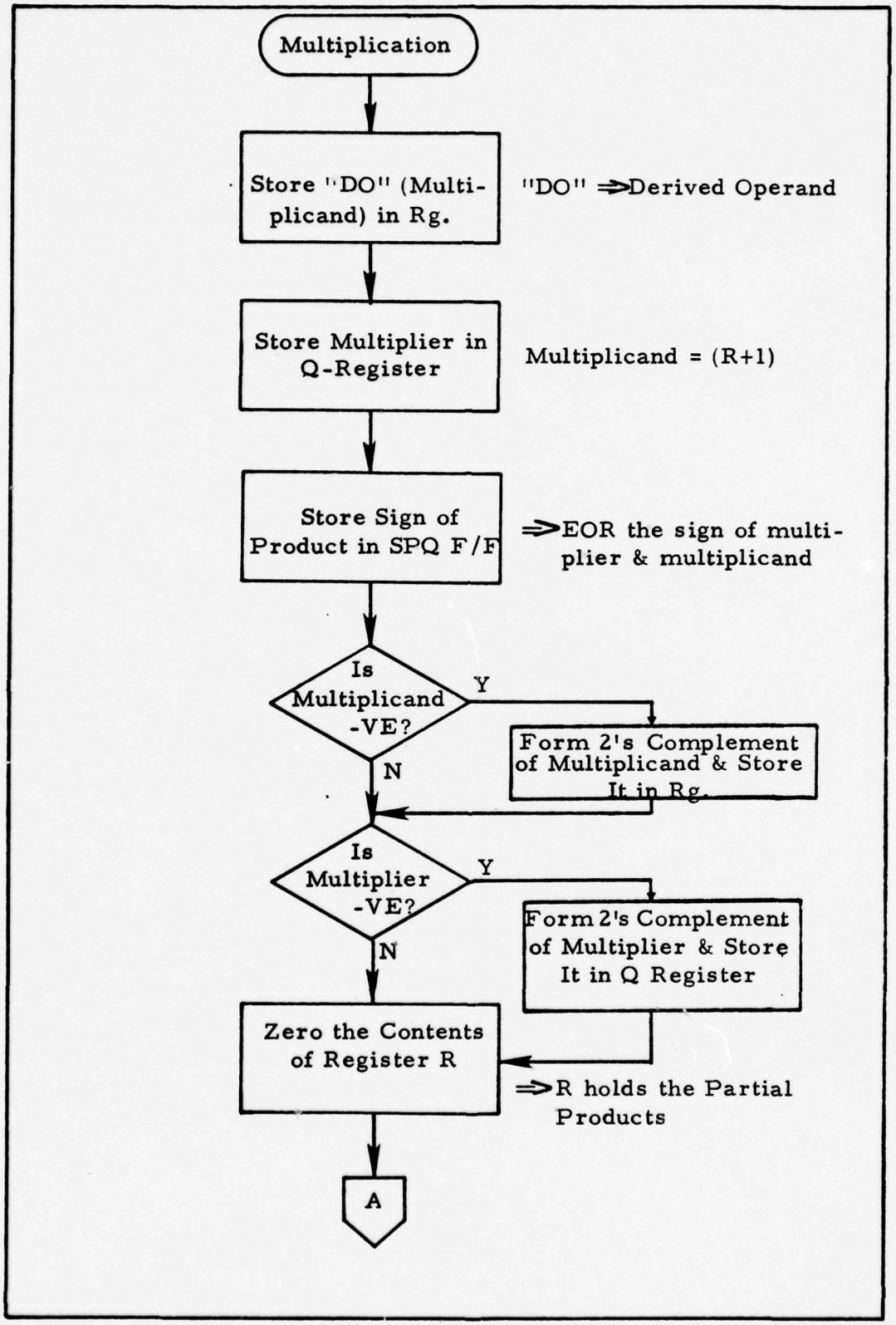


Fig. B-10. Execution Phase--Arithmetic Instructions (Continued)

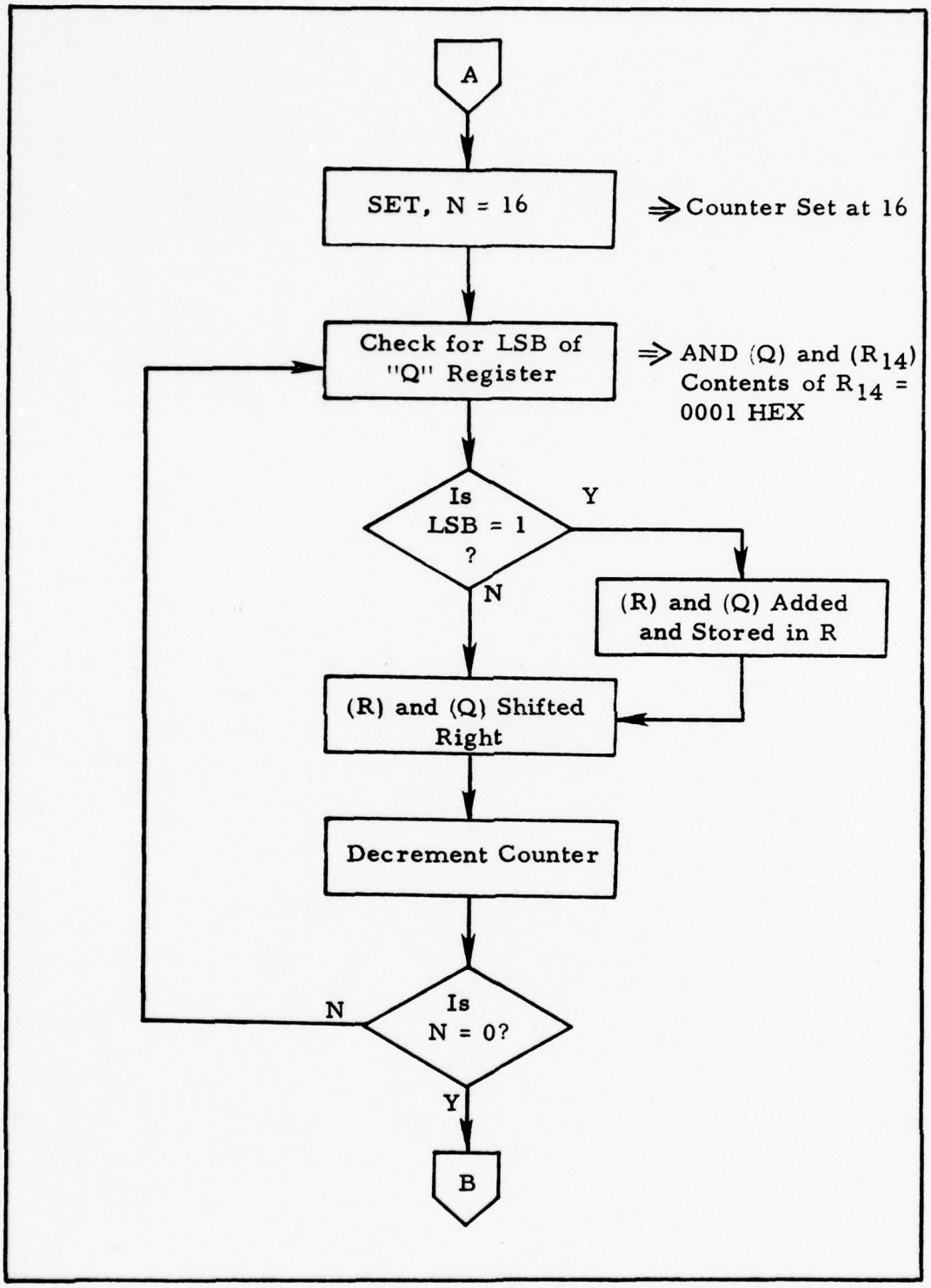


Fig. B-10. Execution Phase--Arithmetic Instructions (Continued)

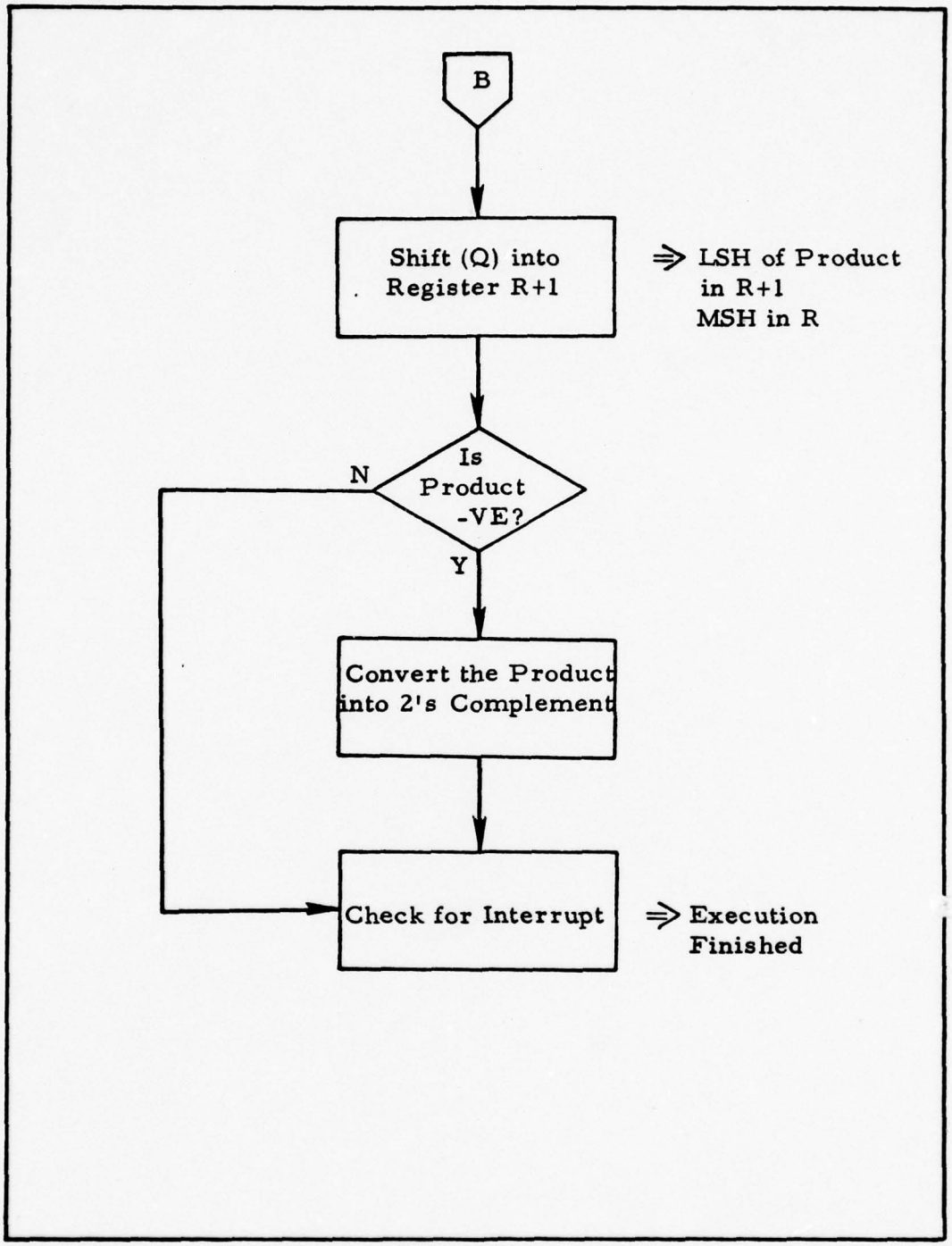


Fig. B-10. Execution Phase--Arithmetic Instructions (Continued)

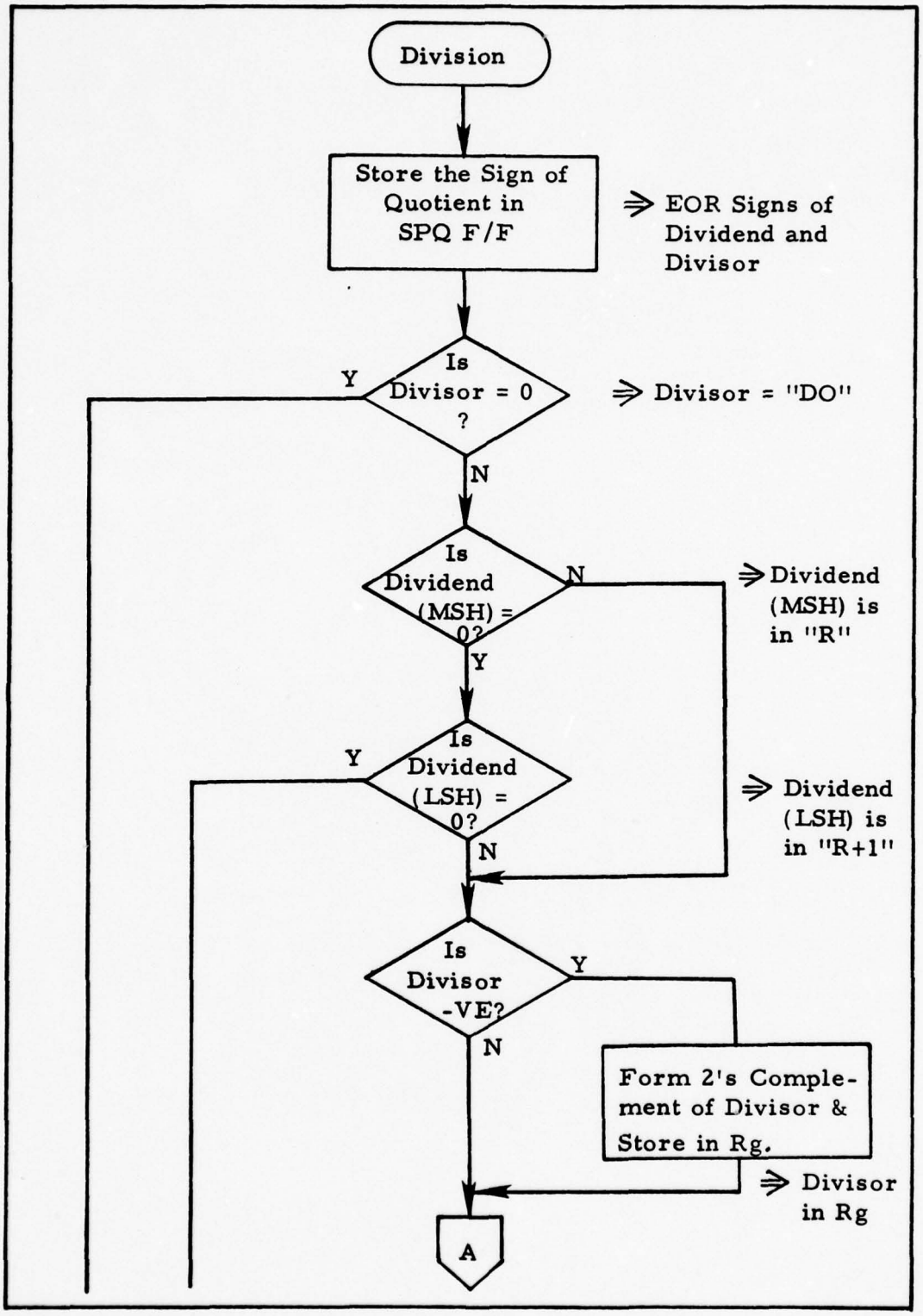


Fig. B-10. Execution Phase--Arithmetic Instructions (Continued)

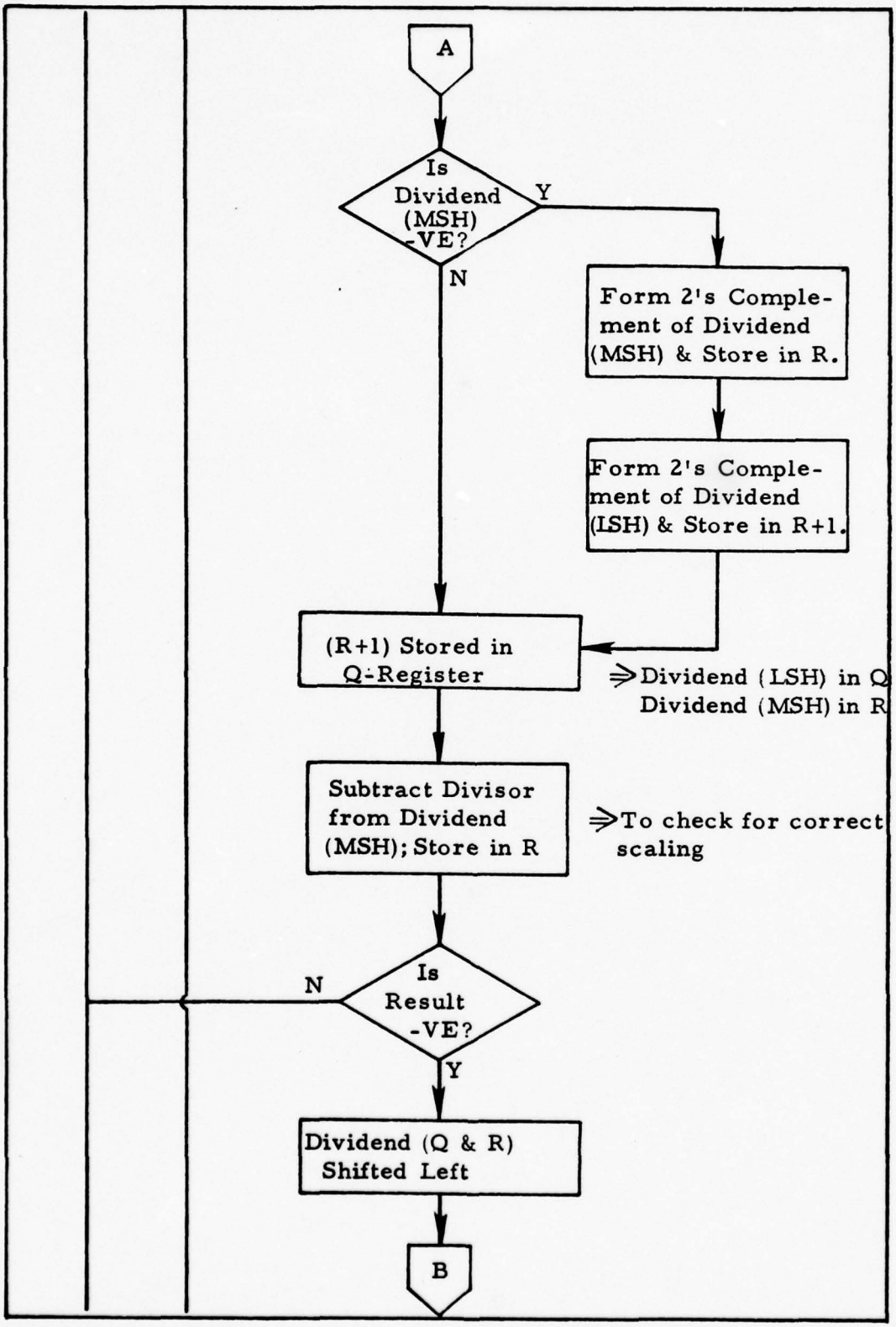


Fig. B-10. Execution Phase--Arithmetic Instructions (Continued)

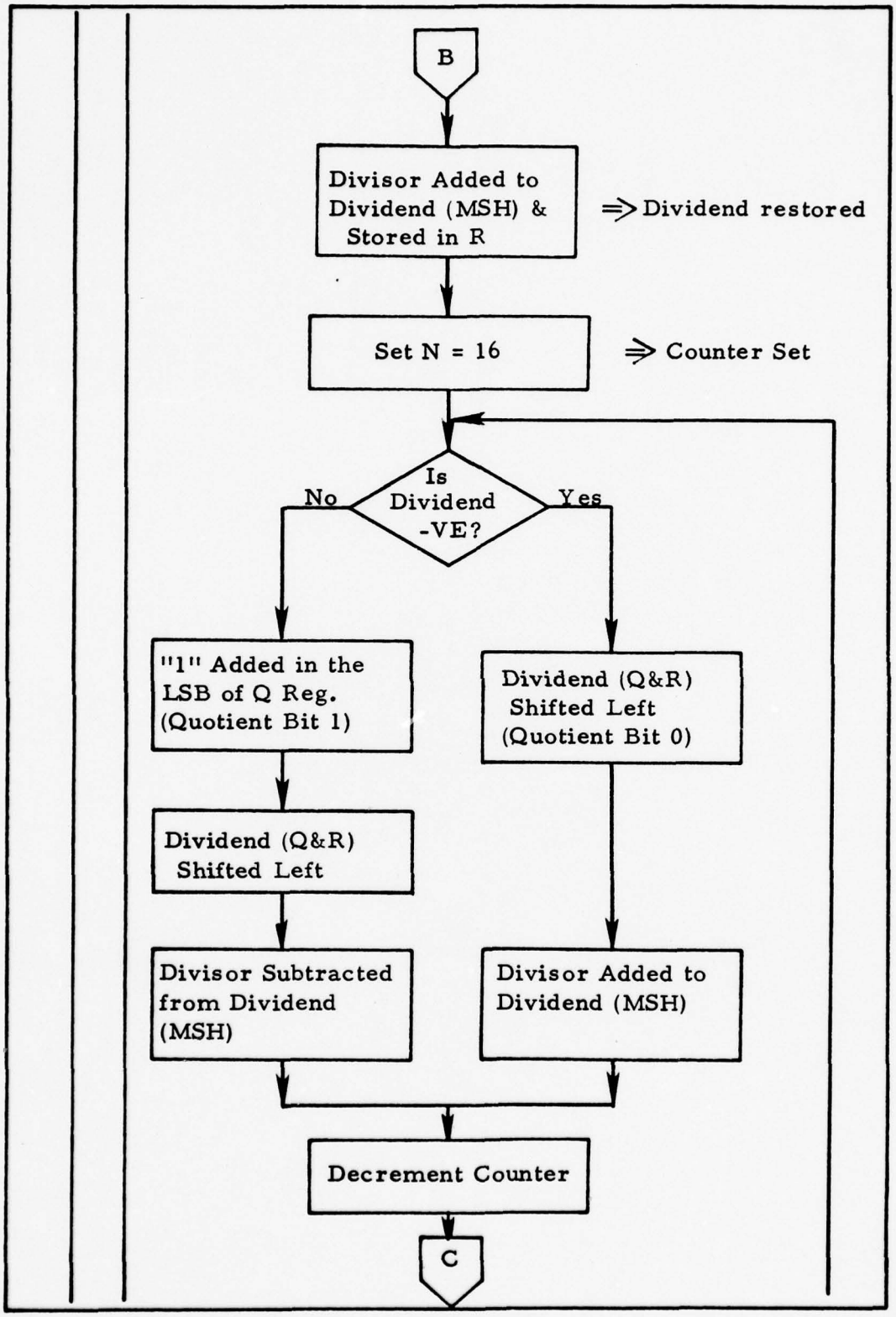


Fig. B-10. Execution Phase--Arithmetic Instructions (Continued)

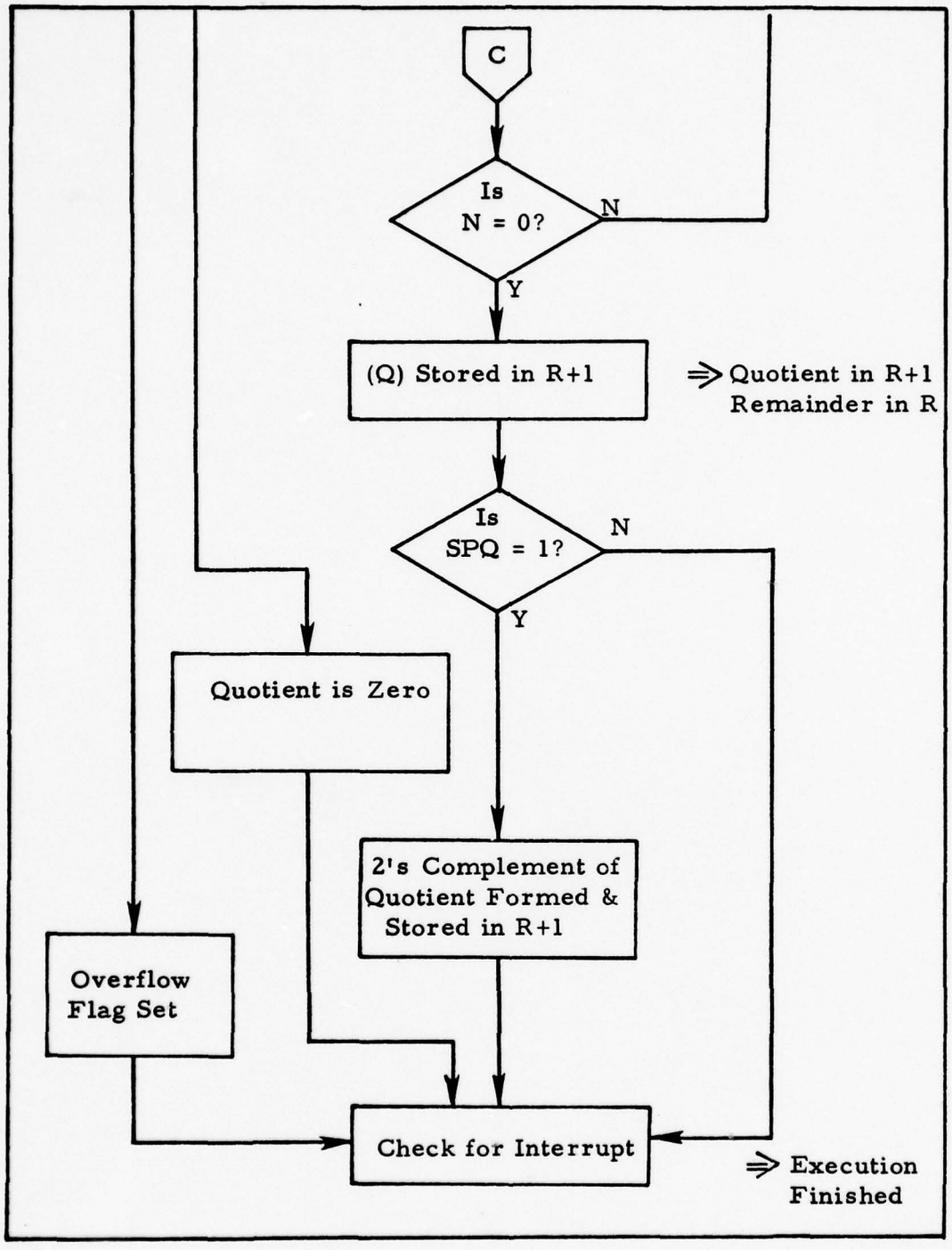


Fig. B-10. Execution Phase--Arithmetic Instructions (Continued)

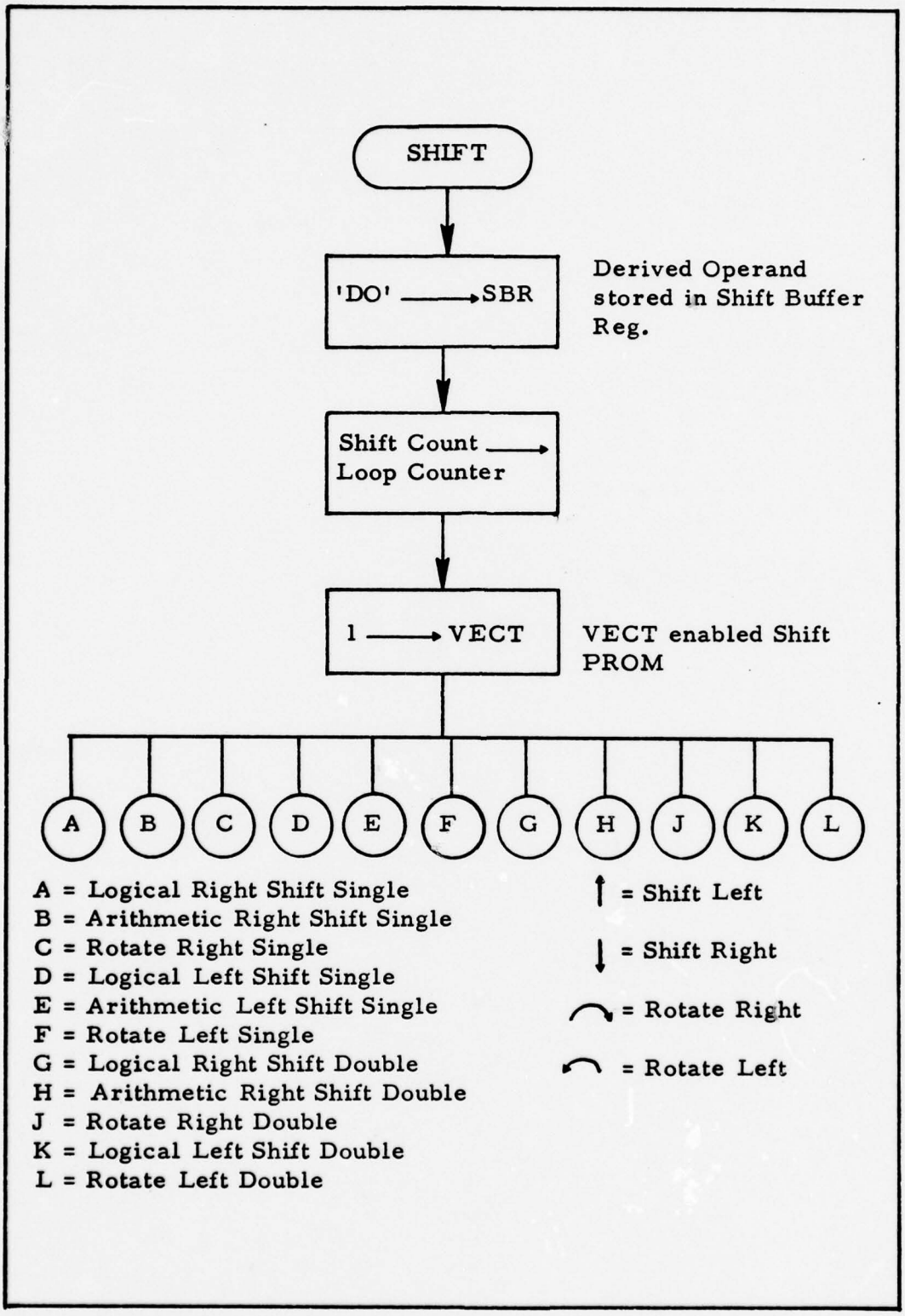


Fig. B-11. Execution Phase--Shift Instructions

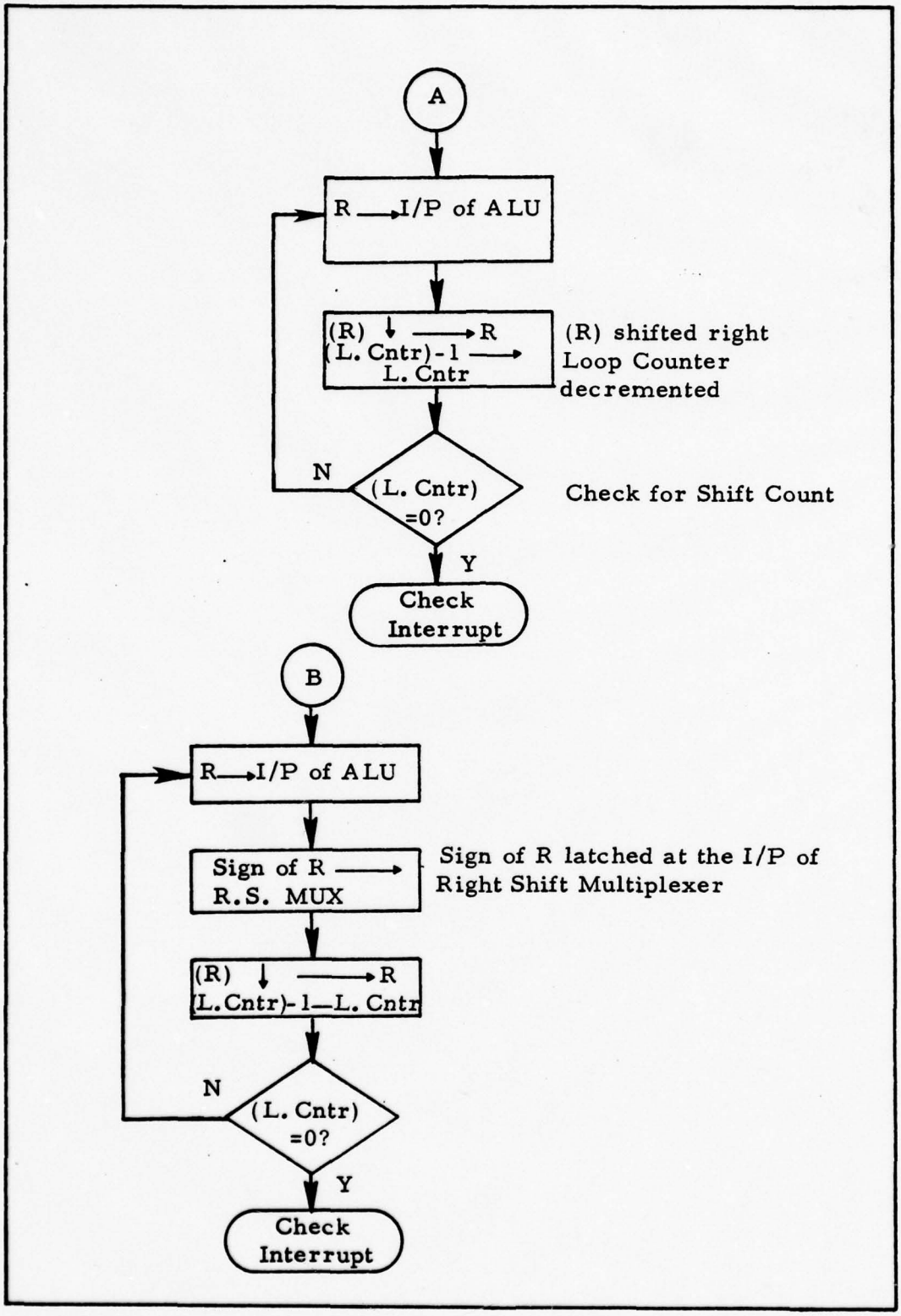


Fig. B-11. Execution Phase--Shift Instructions (Continued)

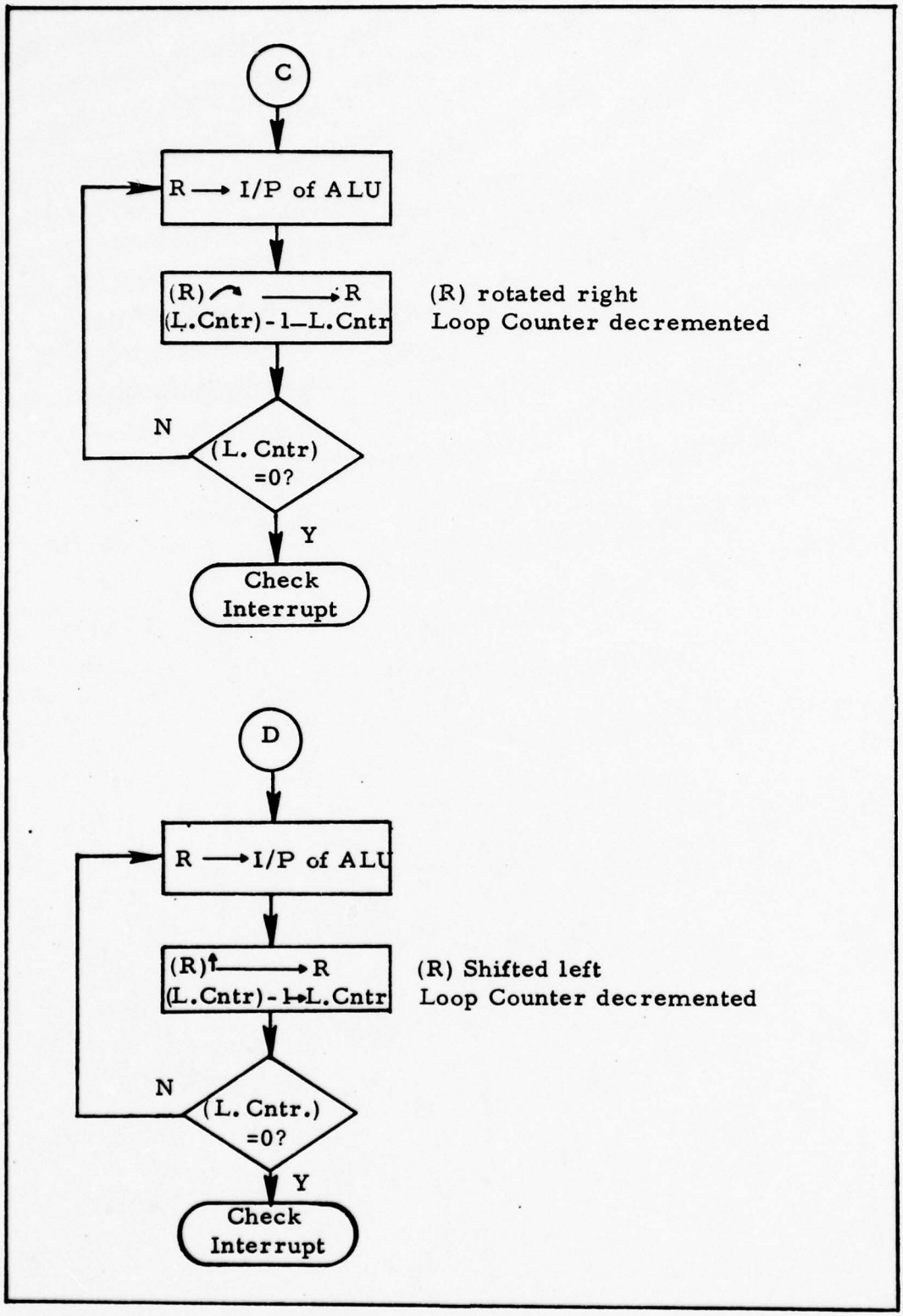


Fig. B-11. Execution Phase--Shift Instructions (Continued)

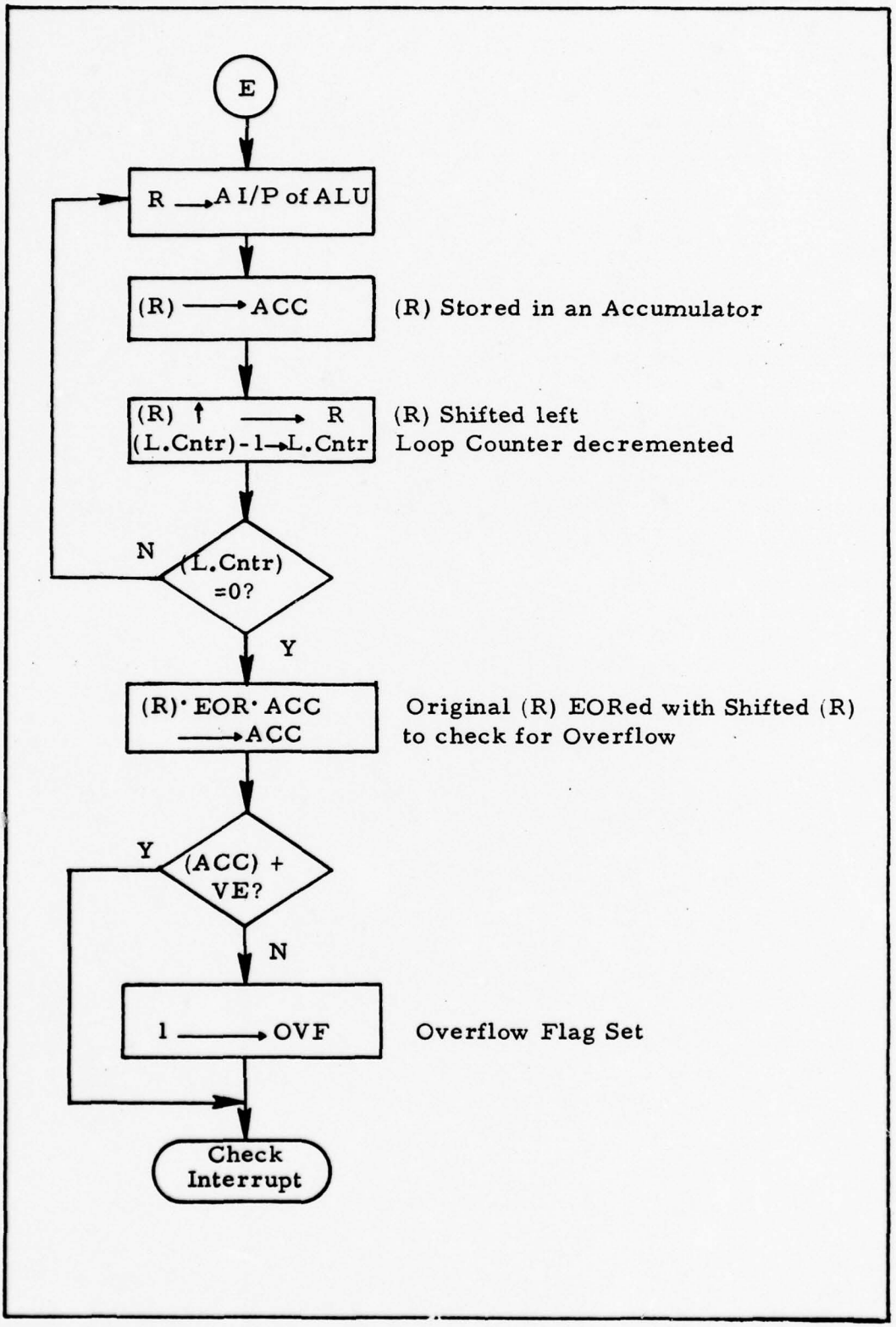


Fig. B-11. Execution Phase--Shift Instructions (Continued)

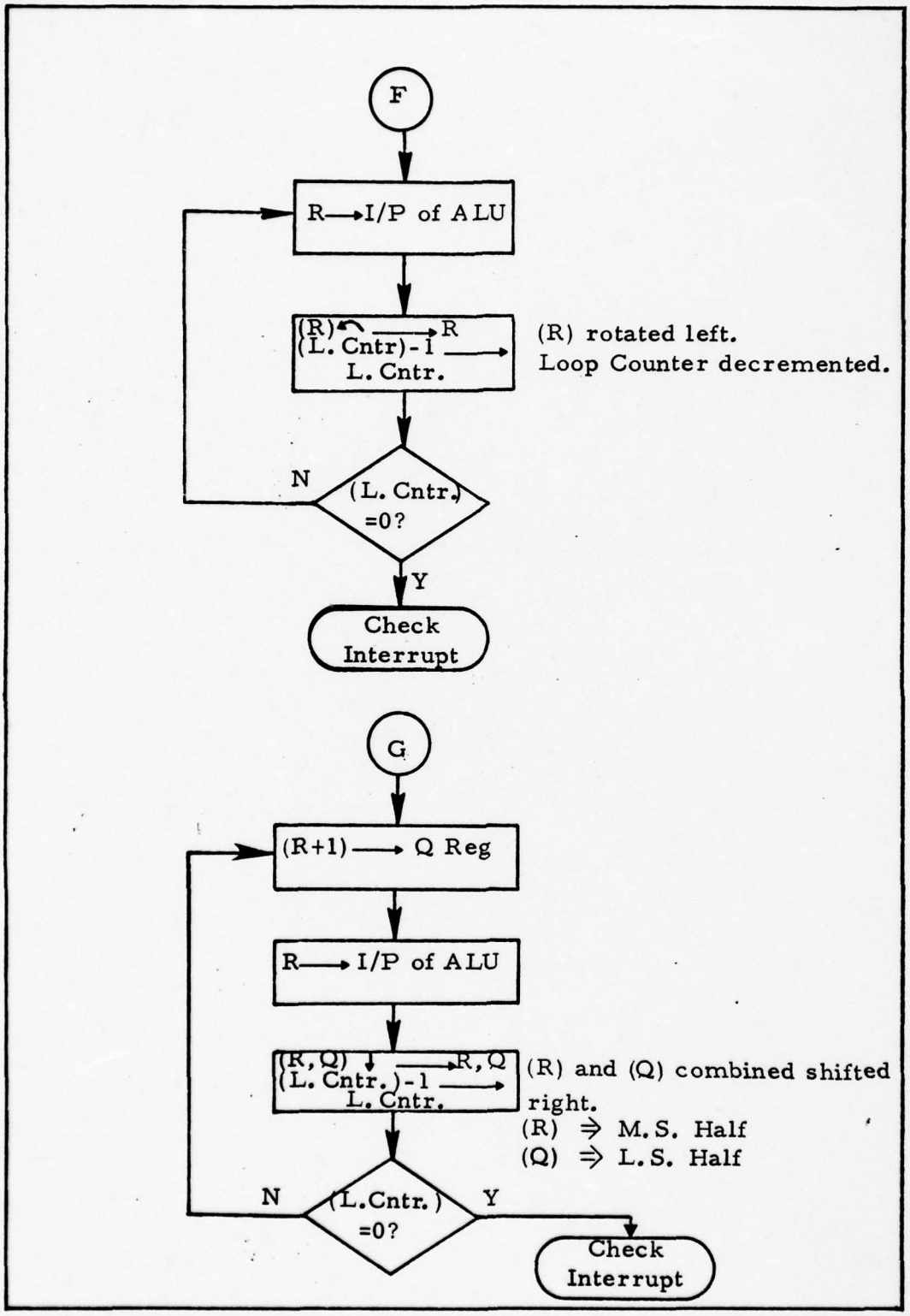


Fig. B-11. Execution Phase--Shift Instructions (Continued)

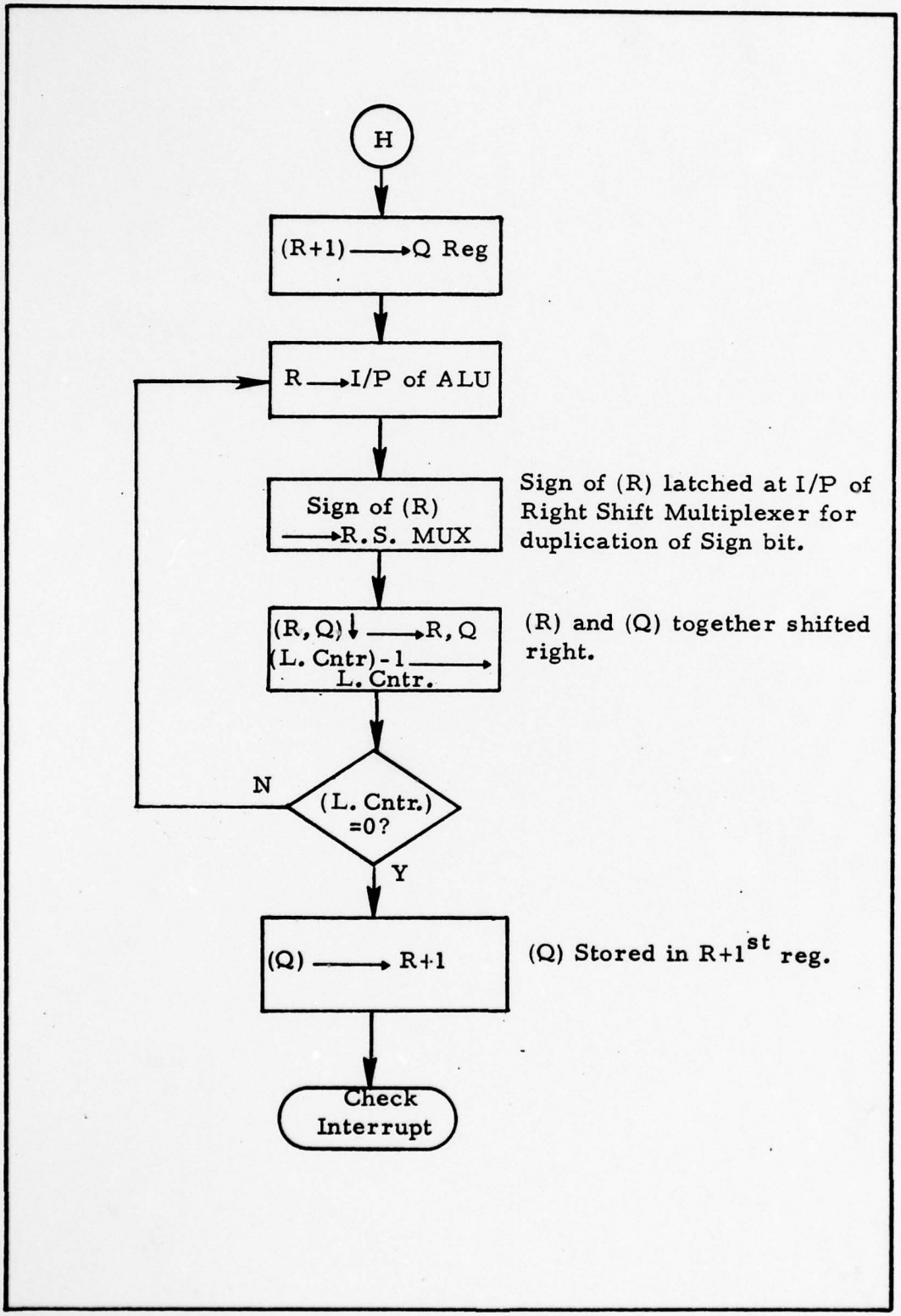


Fig. B-11. Execution Phase--Shift Instructions (Continued)

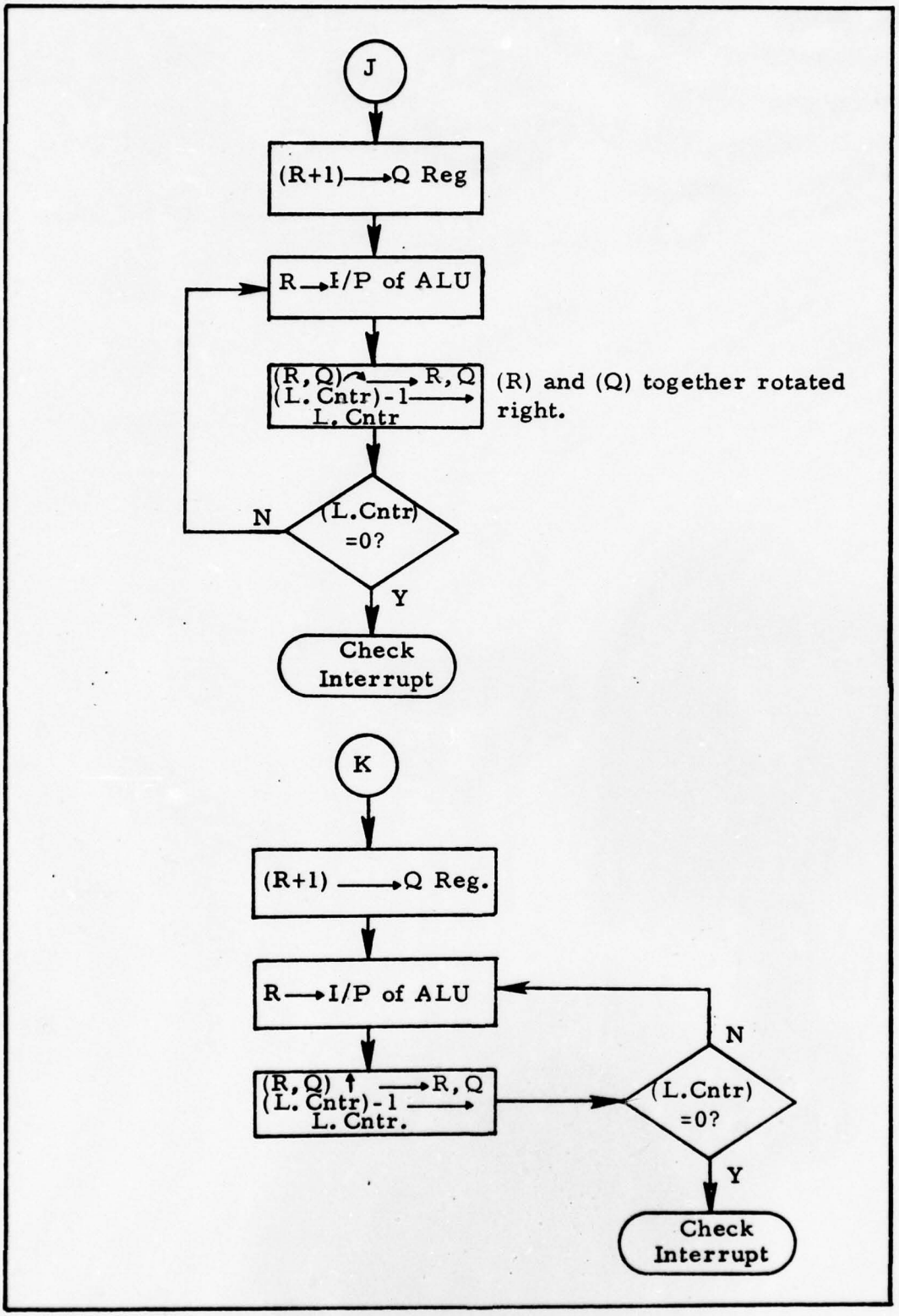


Fig. B-11. Execution Phase--Shift Instructions (Continued)

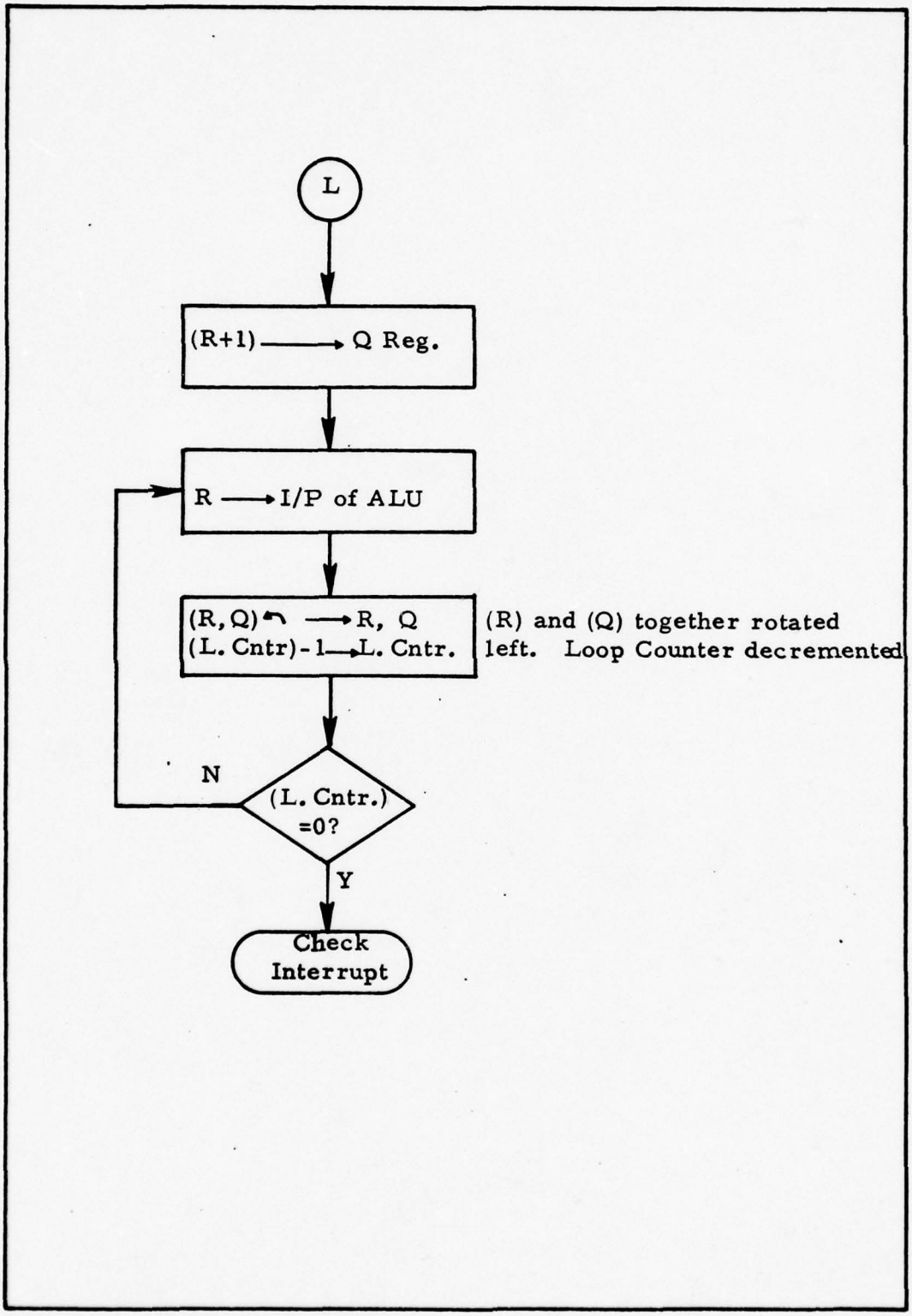


Fig. B-11. Execution Phase--Shift Instructions (Continued)

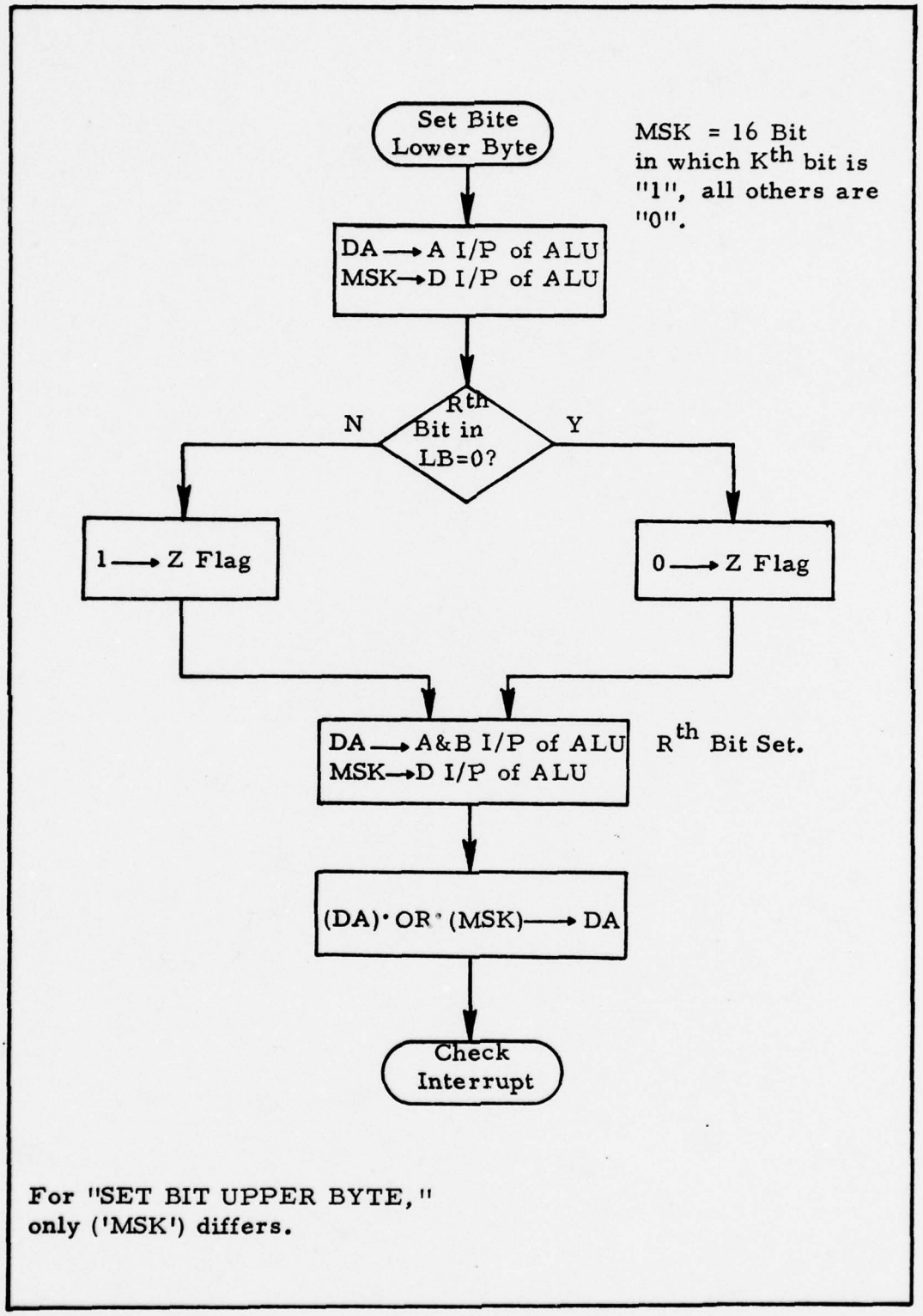


Fig. B-12. Execution Phase--Bit Manipulation Instructions

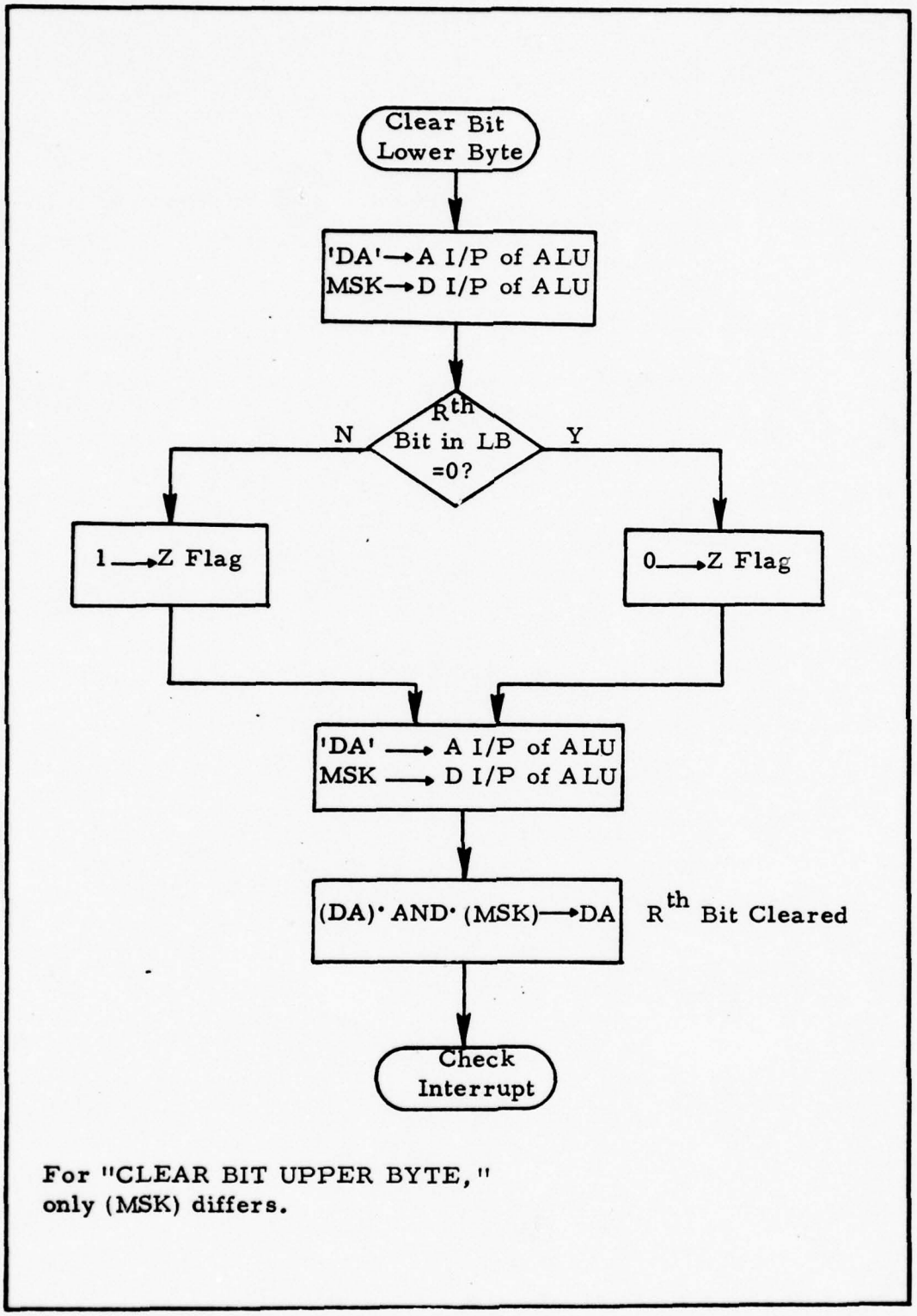


Fig. B-12. Execution Phase--Bit Manipulation Instructions (Cont.)

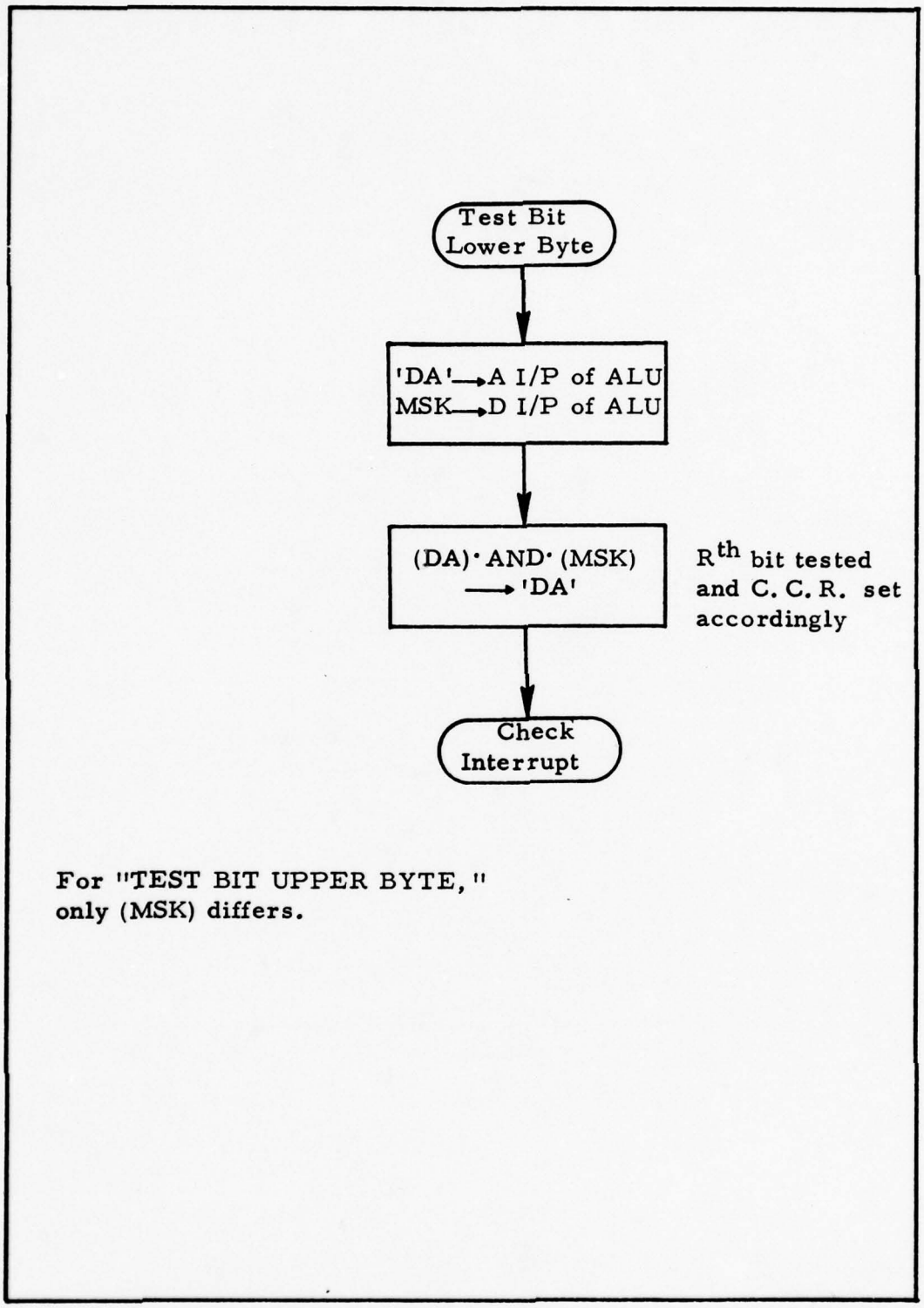


Fig. B-12. Execution Phase--Bit Manipulation Instructions (Cont.)

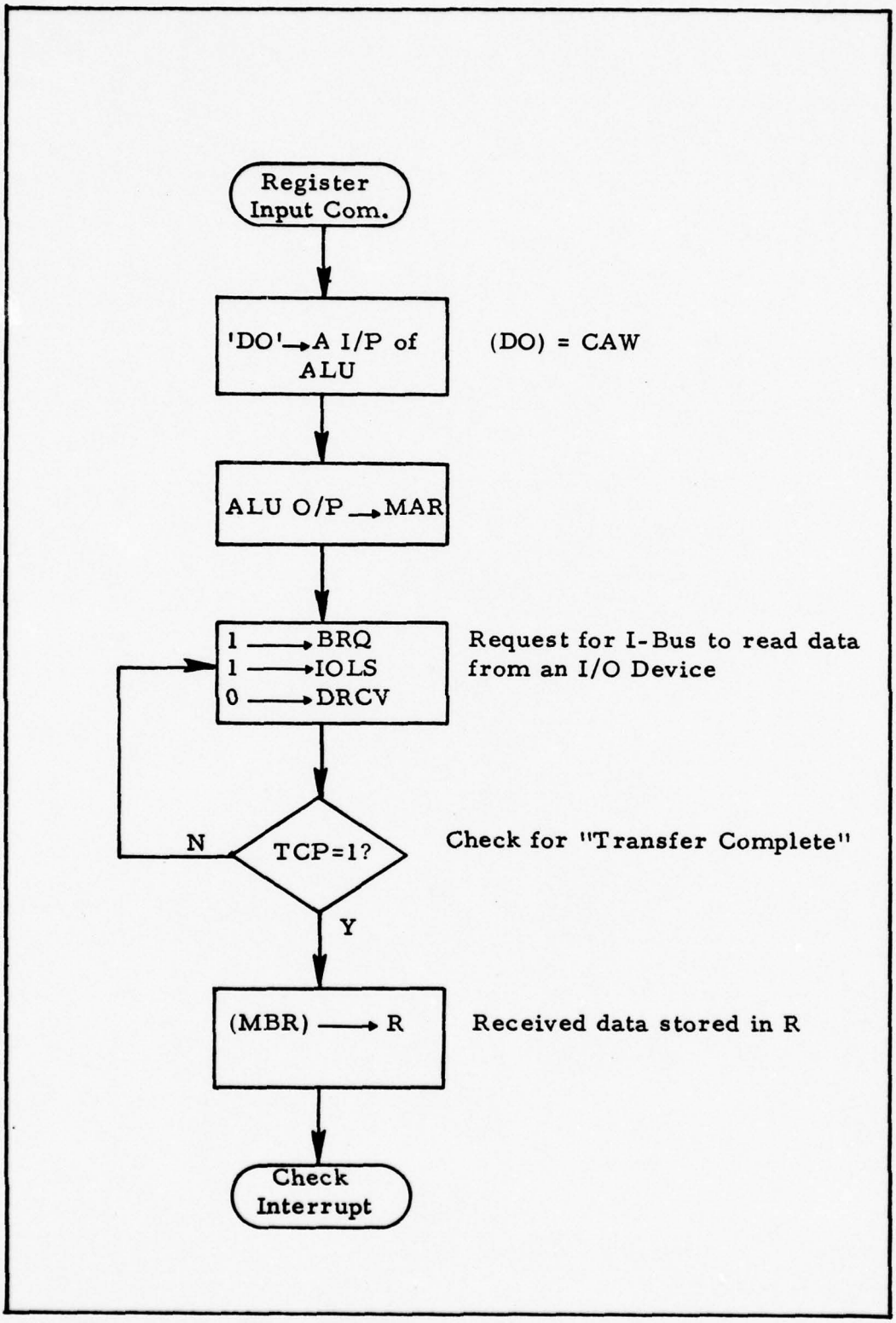


Fig. B-13. Execution Phase--I/O Instructions

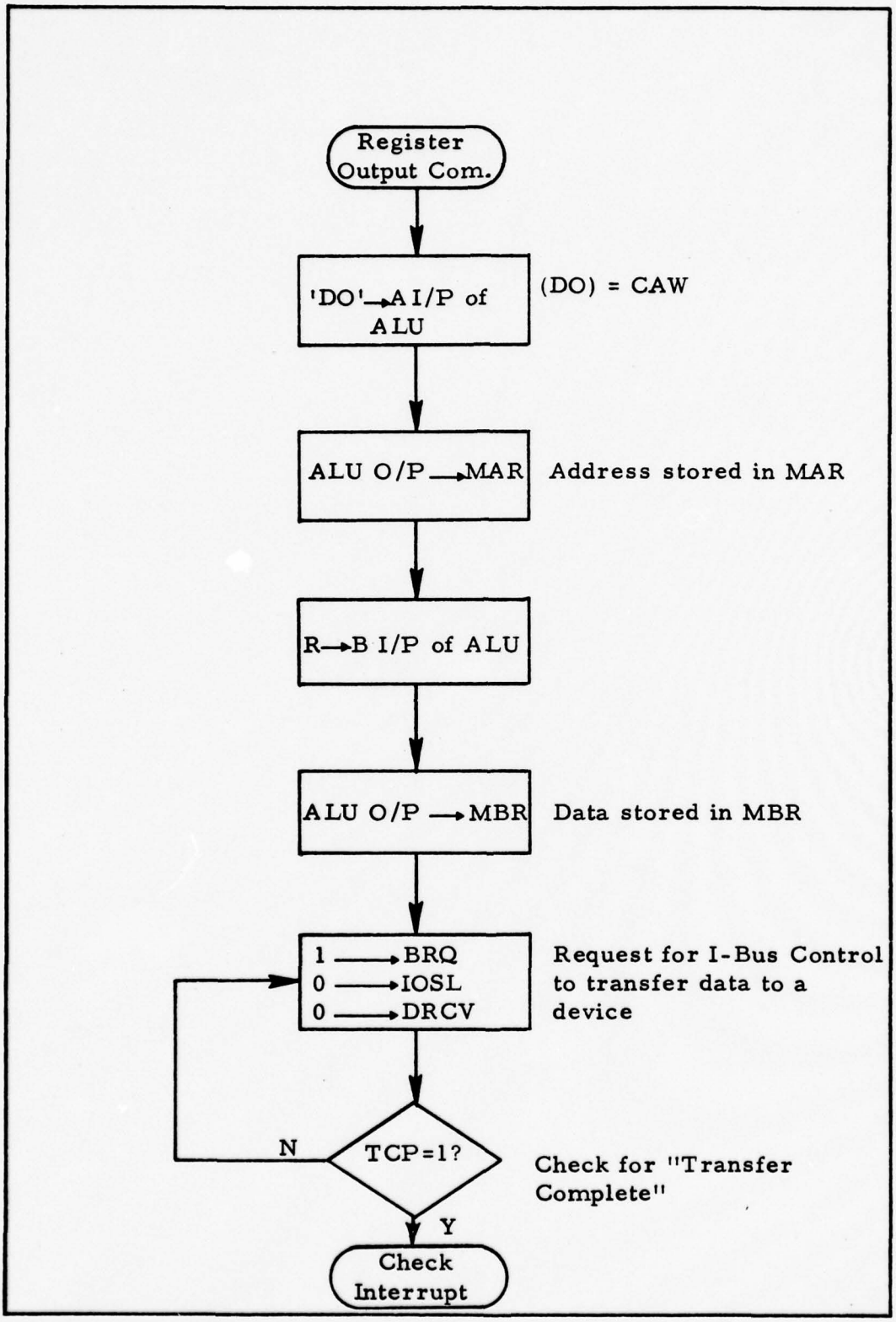


Fig. B-13. Execution Phase--I/O Instructions (Continued)

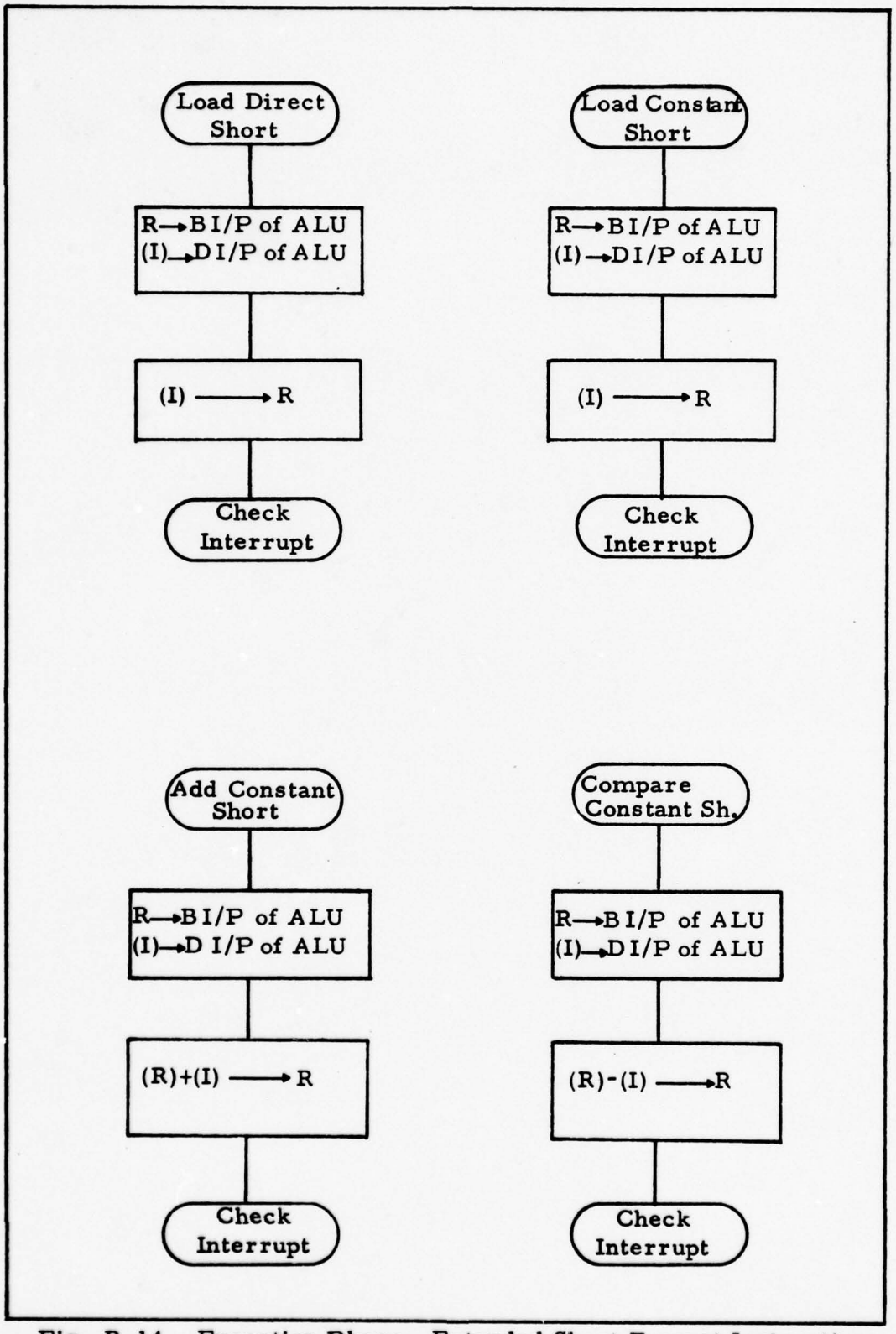


Fig. B-14. Execution Phase--Extended Short Format Instructions

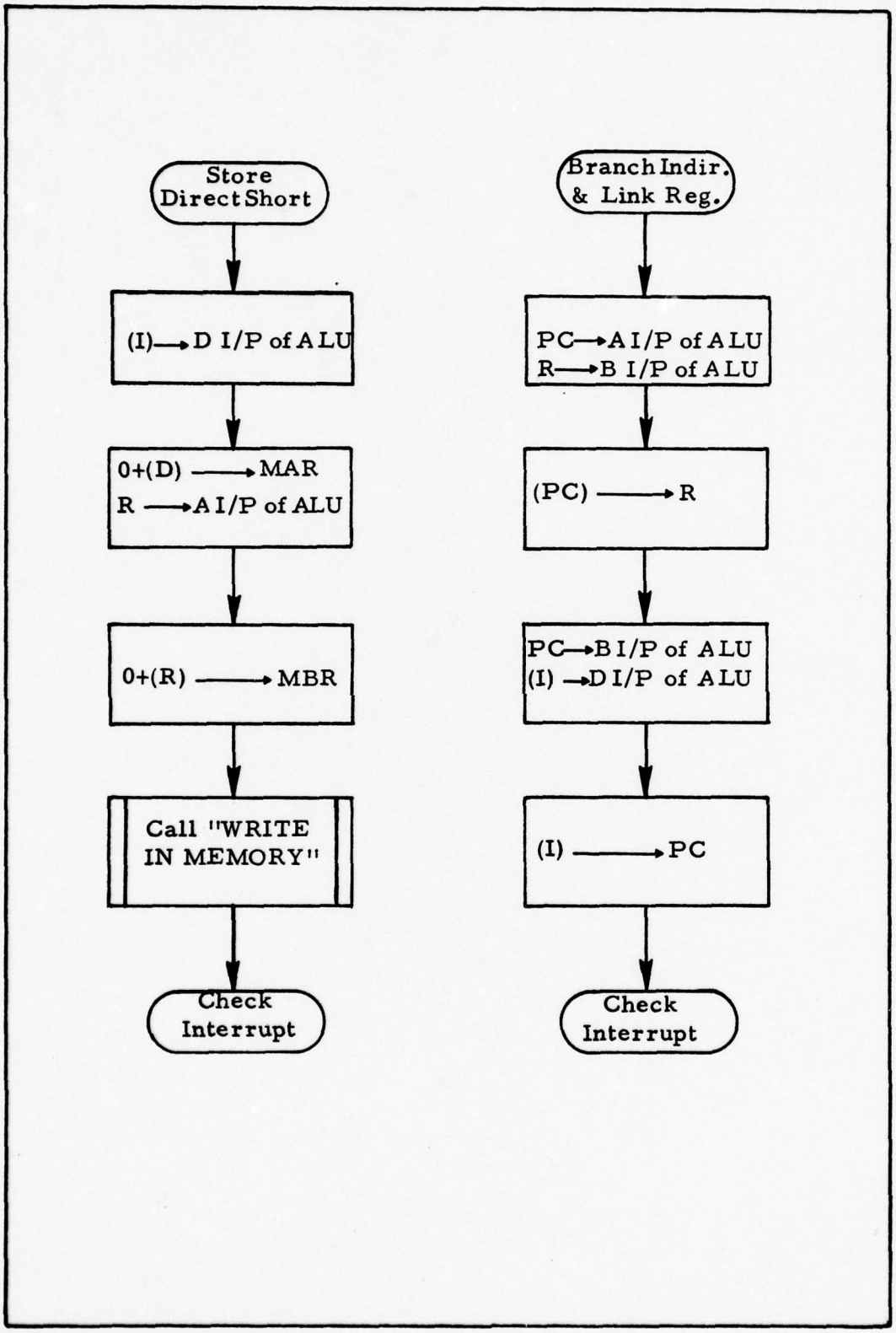


Fig. B-14. Execution Phase--Extended Short Format Instructions (Continued)

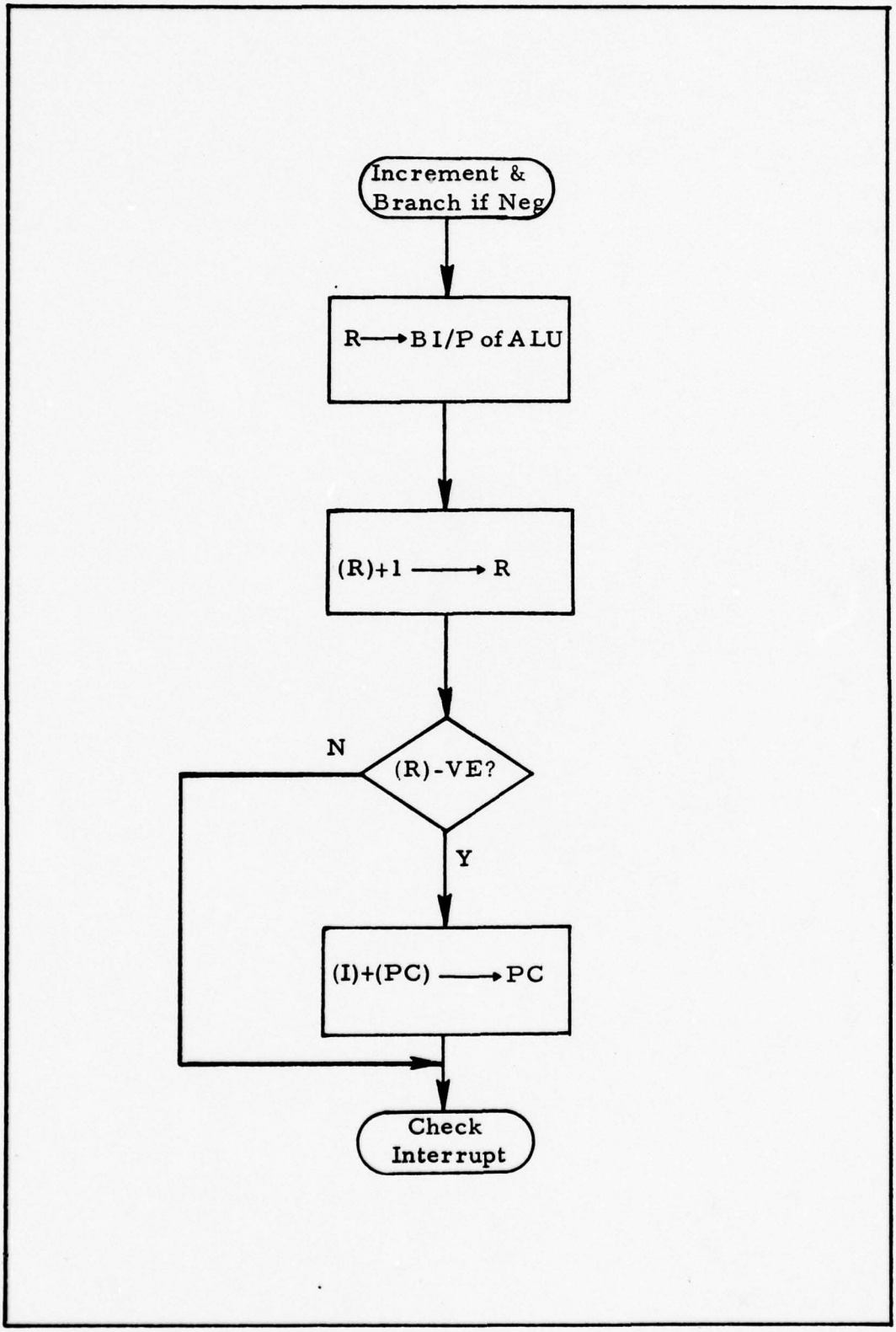


Fig. B-14. Execution Phase--Extended Short Format Instructions (Continued)

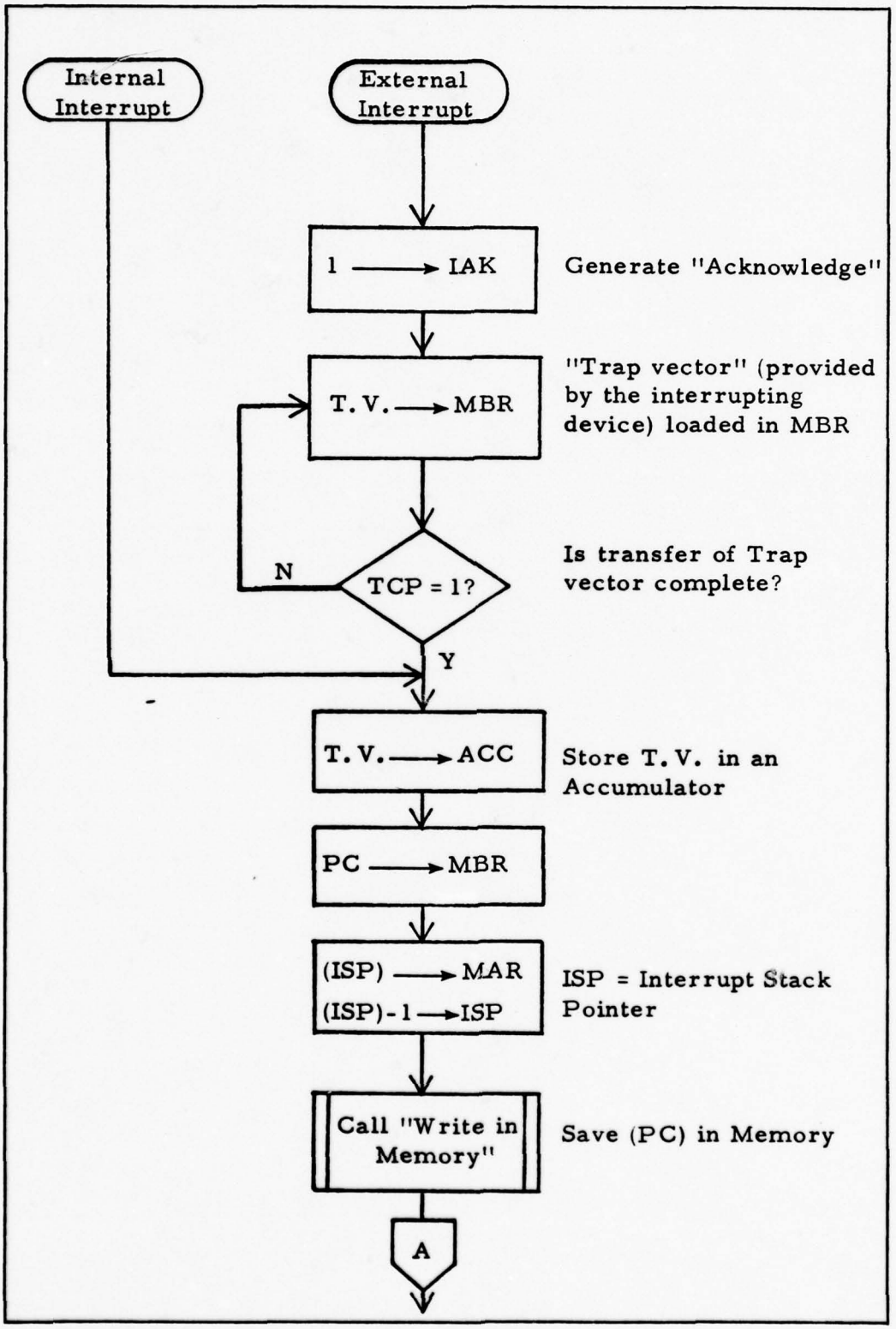


Fig. B-15. Interrupt Handling Routine

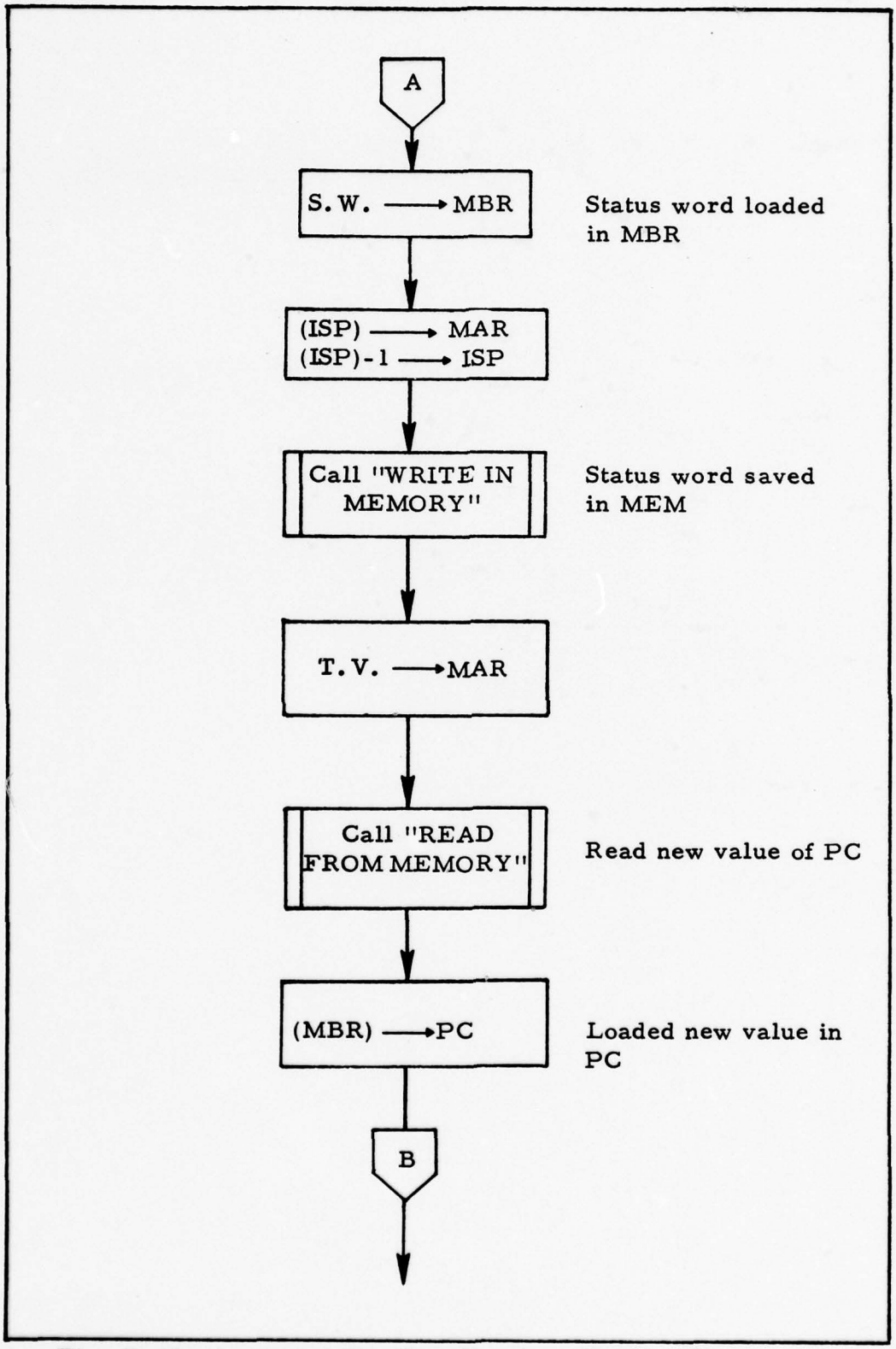


Fig. B-15. Interrupt Handling Routine (Continued)

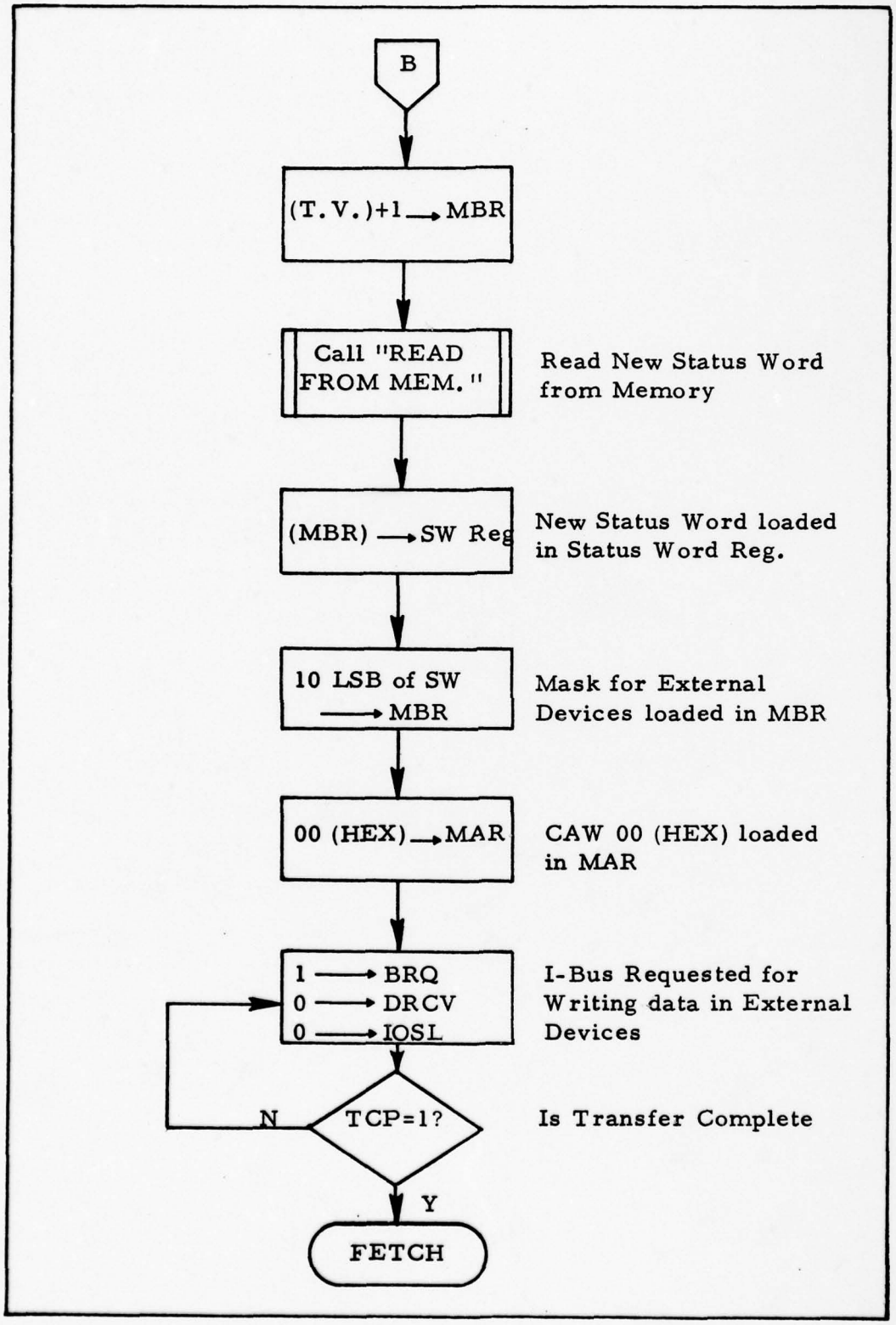


Fig. B-15. Interrupt Handling Routine (Continued)

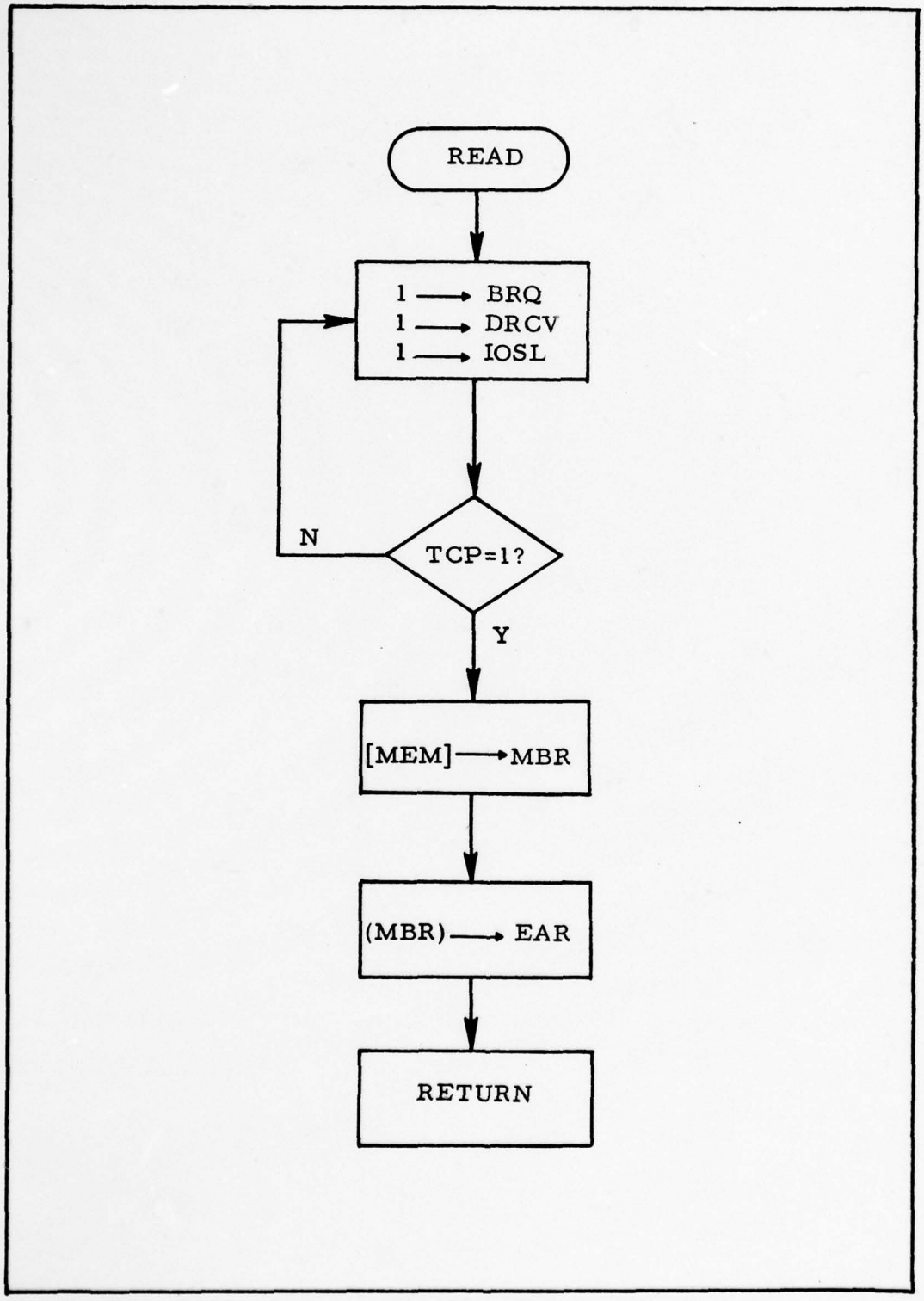


Fig. B-15. Interrupt Handling Routine (Continued)

Appendix C

Micro Codes

Appendix C consists of micro codes for the specified instruction set. The control fields for each microinstruction appear on two consecutive pages facing each other. Both pages carry the address of the microinstruction for ease of reference.

The micro codes have been arranged in the following sequence:

a. "Power-Up" and "Fetch" phase

b. Operand Derivation phase

c. Execution phase

d. Interrupt Handling phase

Table C-1
"Power Up" and Fetch Phase

Function	Address	ALU			C_n	MUX. A	MUX. B	MUX. C	Shift Control	R-Count Control	Enable Control
		S	D	F							
POWER-UP	IN	--	--	--	-	--	LDP	--	--	--	MP
	IN+1	D0	RM	OR	-	--	LDP	--	--	--	MP
	IN+2	0B	RM	AND	-	--	--	--	--	--	MP
FETCH INSTRUCTION	FH	--	--	--	-	--	--	--	--	--	MP
	FH+1	--	--	--	-	--	--	--	--	--	MP
	FH+2	--	--	--	-	LDP	LDP	--	--	--	MP
	FH+3	0B	RA	PLS	1	--	--	--	--	--	MP, AL
	FH+4	--	--	--	-	--	--	--	--	--	MP, IR
	FH+5	--	--	--	-	--	--	AM	--	--	MP

Table C-1 (Continued)
"Power Up" and Fetch Phase

Address	Jump Address	Next Address	Cond. Select	Intpt. Control	Register Control	I-Bus Control			Misc.	Remarks
						Brq	Drcv	Iosl		
IN	--	CON	--	MCL	--	--	--	--	--	Clear all Interrupts.
IN+1	--	CON	--	--	--	--	--	--	CSW	0100 (HEX.) → PC. 0 → SW.
IN+2	FH	CJP	TC	--	--	--	--	--	--	0 → ISP. Go to "FH."
FH	--	CON	--	--	--	--	--	--	--	Continue.
FH+1	FH+2	JRP	HLT	--	--	--	--	--	--	If HALT ≠ 1, go to FH.
FH+2	--	CON	--	--	--	--	--	--	--	PC → A&B.
FH+3	--	CON	--	--	MAI	1	1	1	--	(PC) → MAR. (PC)+1 → PC I-Bus Requested.
FH+4	FH+4	JRP	TCP	--	--	--	--	--	--	Check for Transfer Complete.
FH+5	--	JMA	--	--	--	--	--	--	--	Select Addressing Mode.

Table C-2
Operand Derivation Phase

Function	Address	ALU			C_n	MUX. A	MUX. B	MUX. C	Shift Control	R-Count Control	Enable Control
		S	D	F							
REG. TO REG. MODE	RR	--	--	--	-	--	--	OC1	--	--	MP
REG. INDIRECT MODE	RI	--	--	--	-	LDT	LDP	--	--	--	MP
	RI+1	0A	RM	OR	-	--	--	--	--	--	MP
	RI+2	--	--	--	-	--	--	OC1	--	--	MP
'READ'-SUBROUTINE	RS	--	--	--	-	--	--	--	--	--	MP
	RS+1	--	--	--	-	--	--	--	--	--	MP
	RS+2	--	--	--	-	--	LDP	--	--	--	MP
	RS+3	D0	RM	OR	-	--	--	--	--	--	MP
REG. INDIRECT AUTOINCREMENT MODE	RA	--	--	--	-	LDT	LDP	--	--	--	MP
	RA+1	0A	RM	OR	-	LDT	LDT	--	--	--	MP, AL
	RA+2	0A	RM	PLS	1	--	--	--	--	--	MP

Table C-2 (Continued)
Operand Derivation Phase

Address	Jump Address	Next Address	Cond. Select	Intpt. Control	Register Control	I-Bus Control			Misc.	Remarks
						Brq	Drcv	Josl		
RR	--	JMA	--	--	--	--	--	--	--	Select OP-Code C1 for execution.
RI	--	CON	--	--	--	--	--	--	--	T→A, EAR→B.
RI+1	RS	CSP	AO	--	MAI	--	--	--	--	(T)→EAR, If AO=1 Call "READ."
RI+2	--	JMA	--	--	--	--	--	--	--	Select Op-Code C1 for execution.
RS	--	CON	--	--	--	1	1	1	--	Request for I-Bus.
RS+1	RS+2	JRP	TCP	--	--	--	--	--	--	If TCP=1, go to RS+2.
RS+2	--	CON	--	--	MBI	--	--	--	--	(MEM.)→MBR, EAR→B.
RS+3	--	RTN	TC	--	--	--	--	--	--	(MBR)→EAR.
RA	RA+4	CJP	T=7	--	--	--	--	--	--	If T=7, go to RA+4.
RA+1	--	CON	--	--	MAI	--	--	--	--	(T)→EAR&MAR, T→A&B.
RA+2	RS	CSP	AO	--	--	--	--	--	--	(T)+1→T. If AO=1, Call "READ."

Table C-2 (Continued)
Operand Derivation Phase

Function	Address	ALU			C_n	MUX. A	MUX. B	MUX. C	Shift Control	R-Count Control	Enable Control
		S	D	F							
REG. INDIRECT AUTOINCREMENT MODE (Continued)	RA+3	--	--	--	-	--	--	OC1	--	--	MP
	RA+4	--	--	--	-	LDP	LDP	--	--	--	MP
	RA+5	0A	RA	PLS	1	--	--	--	--	--	MP, AL
	RA+6	--	--	--	-	--	--	--	--	--	MP
DIRECT AND DIRECT-INDEXED MODE	DI	--	--	--	-	LDP	LDP	--	--	--	MP
	DI+1	0A	RA	PLS	1	--	--	--	--	--	MP, AL
	DI+2	--	--	--	-	LDT	LDP	--	--	--	MP
	DI+3	AB	RM	PLS	0	--	--	--	--	--	MP
	DI+4	--	--	--	-	--	--	OC1	--	--	MP
EXTENDED SHORT FORMAT MODE	ES	--	--	--	-	--	--	--	--	--	MP
	ES+1	--	--	--	-	LDP	LDP	OC2	--	--	MP
	ES+2	0A	RA	PLS	1	--	--	--	--	--	MP, AL

Table C-2 (Continued)
Operand Derivation Phase

Address	Jump Address	Next Address	Cond. Select	Intpt. Control	Register Control	I-Bus Control			Misc.	Remarks
						Brq	Drcv	Iosl		
RA+3	--	JMA	--	--	--	--	--	--	--	AO≠1, Select OP-Code C1
RA+4	--	CON	--	--	--	--	--	--	--	PC→A&B.
RA+5	RS	CSP	TC	--	MAI	--	--	--	--	(PC)→MAR. Call "READ"
RA+6	RA+3	CJP	TC	--	--	--	--	--	--	Go to RA+3.
DI	--	CON	--	--	--	--	--	--	--	PC→A&B.
DI+1	RS	CSP	TC	--	MAI	--	--	--	--	(PC)→MAR, PC+1→PC. Call "READ."
DI+2	--	CON	--	--	--	--	--	--	--	T→A, EAR→B.
DI+3	RS	CSP	AO	--	--	--	--	--	--	(T)+(EAR)→EAR. Call "READ."
DI+4	--	JMA	--	--	--	--	--	--	--	Select OP-Code C1.
ES	ES+2	CJP	MIO	--	LIR	--	--	--	--	Load I-field Reg. If MIO=1, go to ES+2.
ES+1	--	JMA	--	--	--	--	--	--	--	Select OP-Code C2.
ES+2	RS	CSP	TC	--	MAI	--	--	--	--	(PC)→MAR, Call "READ"

Table C-2 (Continued)
Operand Derivation Phase

Function	Address	ALU			C_n	MUX. A	MUX. B	MUX. C	Shift Control	R-Count Control	Enable Control
		S	D	F							
EXTENDED SHORT FORMAT MODE (Continued)	ES+3	--	--	--	-	LDT	LDP	--	--	--	MP
	ES+4	AB	RM	PLS	0	--	--	--	--	--	MP
LOAD/STORE MODE	LS	--	--	--	-	--	--	--	--	--	MP
	LS+1	--	--	--	-	LDT	LDP	--	--	--	MP
	LS+2	DA	RM	PLS	0	--	--	OC2	--	--	MP

Table C-2 (Continued)
Operand Derivation Phase

Address	Jump Address	Next Address	Cond. Select	Intpt. Control	Register Control	I-Bus Control			Misc.	Remarks
						Brq	Drcv	Iosl		
ES+3	CON	--	--	--	--	--	--	--	--	T→A and EAR→B.
ES+4	ES+1	CJP	TC	--	--	--	--	--	--	Go to (ES+1).
LS	--	CON	--	--	LIR	--	--	--	--	Load I-field Reg.
LS+1	--	CON	--	--	SIO	--	--	--	--	I(Sign extended)→D, EAR →B.
LS+2	--	JMA	--	--	--	--	--	--	--	(T)+(I)→EAR. Select OP-Code C2.

Table C-3
Execution Phase--Data Transfer Instructions

Function	Address	ALU			C_n	MUX. A	MUX. B	MUX. C	Shift Control	R-Count Control	Enable Control
		S	D	F							
LOAD	LD	--	--	--	-	LDP	LDR	--	--	--	MP
	LD+1	0A	RM	OR	-	--	--	--	--	--	MP
STORE	ST	--	--	--	-	LDP	--	--	--	--	MP
	ST+1	0A	F	OR	-	--	LDR	--	--	--	MP, AL
	ST+2	0B	F	OR	-	--	--	--	--	--	MP, AL
	ST+3	--	--	--	-	--	--	--	--	--	MP
STORE THROUGH MASK	SM	--	--	--	-	LDP	--	--	--	--	MP
	SM+1	0A	F	OR	-	--	--	--	--	IRC	MP, AL
	SM+2	--	--	--	-	LDR	LDC	--	--	--	MP
	SM+3	AB	F	AND	-	--	--	--	--	--	MP, AL
	SM+4	--	--	--	-	--	--	--	--	--	MP
PUSH	PS	--	--	--	-	LDR	LDR	--	--	--	MP
	PS+1	0A	RM	MIN	1	--	LDP	--	--	--	MP, AL

Table C-3 (Continued)
Execution Phase--Data Transfer Instructions

Address	Jump Address	Next Address	Cond. Select	Intpt. Control	Register Control	I-Bus Control			Misc.	Remarks
						Brq	Drcv	Iosl		
LD	--	CON	--	--	--	--	--	--	--	'DO'→A, R→B
LD+1	INC	CJP	TC	--	--	--	--	--	--	(DO)→B, Check interrupt
ST	--	CON	--	--	--	--	--	--	--	'DA'→A
ST+1	--	CON	--	--	MAI	--	--	--	--	(DA)→MAR, R→B
ST+2	WIM	CSP	TC	--	MBI	--	--	--	--	(R)→MBR, Call "WRITE"
ST+3	INC	CJP	TC	--	--	--	--	--	--	Check interrupt
SM	--	CON	--	--	--	--	--	--	--	'DA'→A
SM+1	--	CON	--	--	MAI	--	--	--	--	(DA)→MAR, R-counter incremented
SM+2	--	CON	--	--	--	--	--	--	--	R→A, R+1→B
SM+3	WIM	CSP	TC	--	MBI	--	--	--	--	(R)·AND·(R+1)→MBR·Call "WRITE"
SM+4	INC	CJP	TC	--	--	--	--	--	--	Check interrupt
PS	--	CON	--	--	--	--	--	--	--	R→A, R→B
PS+1	--	CON	--	--	MAI	--	--	--	--	(DO)→MBR, Call "WRITE"

Table C-3 (Continued)
Execution Phase--Data Transfer Instructions

Function	Address	ALU			C_n	MUX. A	MUX. B	MUX. C	Shift Control	R-Count Control	Enable Control
		S	D	F							
PUSH (Continued)	PS+2	0B	F	OR	-	--	--	--	--	--	MP, AL
	PS+3	--	--	--	-	--	--	--	--	--	MP
	PS+4										
MOVE & AUTO-INCREMENT	MA	--	--	--	-	LDP	--	--	--	--	MP
	MA+1	0A	F	OR	-	--	--	--	--	--	MP, AL
	MA+2	--	--	--	-	LDR	LDR	--	--	--	MP
	MA+3	AB	RA	PLS	1	--	--	--	--	--	MP, AL
	MA+4	--	--	--	-	--	--	--	--	--	MP
PUSH MULTIPLE	PU	--	--	--	-	LDR	--	--	--	--	MP, ET
	PU+1	0A	F	OR	-	--	--	--	--	DRC	MP, AL
	PU+2	--	--	--	-	LDP	LDP	--	--	--	MP
	PU+3	AB	RA	MIN	1	--	--	--	--	--	MP, AL
	PU+4	--	--	--	-	--	--	--	--	--	MP

Table C-3 (Continued)
Execution Phase--Data Transfer Instructions

Address	Jump Address	Next Address	Cond. Select	Intpt. Control	Register Control	I-Bus Control			Misc.	Remarks
						Brq	Drcv	Iosl		
PS+2	WIM	CSP	TC	--	MBI	--	--	--	--	(DO)→MBR, Call "WRITE"
PS+3	INC	CJP	TC	--	--	--	--	--	--	Check interrupt
PS+4										
MA	--	CON	--	--	--	--	--	--	--	'DO'→A
MA+1	--	CON	--	--	MBI	--	--	--	--	(DO)→MBR
MA+2	--	CON	--	--	--	--	--	--	--	R→A, R→B
MA+3	WIM	CSP	TC	--	MAI	--	--	--	--	(R)→MAR, (R)+1→R
MA+4	INC	CJP	TC	--	--	--	--	--	--	Check interrupt
PU	--	CON	--	--	--	--	--	--	$\overline{\text{RLD}}$	R→A, (T)→Loop Counter
PU+1	--	CON	--	--	MBI	--	--	--	--	(R)→MBR, R-counter decremented
PU+2	--	CON	--	--	--	--	--	--	--	ISP→A&B
PU+3	WIM	CSP	TC	--	MAI	--	--	--	--	(ISP)→MAR, Call "WRITE"
PU+4	PU	RPC	--	--	--	--	--	--	--	Loop counter ≠ 0, go to PU

Table C-3 (Continued)
Execution Phase--Data Transfer Instructions

Function	Address	ALU			C_n	MUX. A	MUX. B	MUX. C	Shift Control	R-Count Control	Enable Control
		S	D	F							
PUSH MULTIPLE (Continued)	PU+5	--	--	--	-	--	--	--	--	--	MP
POP MULTIPLE	PO	--	--	--	-	LDP	LDP	--	--	--	MP, ET
	PO+1	AB	RA	PLS	1	--	--	--	--	--	MP, AL
	PO+2	--	--	--	-	--	LDR	--	--	--	MP
	PO+3	D0	RM	--	-	--	--	--	--	IRC	MP, AL
	PO+4	--	--	--	-	--	--	--	--	--	MP
	PO+5	--	--	--	-	--	--	--	--	--	MP

Table C-3 (Continued)
Execution Phase--Data Transfer Instructions

Address	Jump Address	Next Address	Cond. Select	Intpt. Control	Register Control	I-Bus Control			Misc.	Remarks
						Brq	Drcv	Iosl		
PU+5	INC	CJP	--	--	--	--	--	--	--	Loop Counter = 0, check intpt.
PO	--	CON	--	--	--	--	--	--	--	ISP→A&B, (T)→Loop Counter
PO+1	RFM	CSP	TC	--	MAI	--	--	--	--	(ISP)→MAR, Call "READ"
PO+2	--	CON	--	--	--	--	--	--	--	R→B
PO+3	--	CON	--	--	--	--	--	--	--	(MEM)→R, R-Counter incremented
PO+4	PO	RPC	--	--	--	--	--	--	--	Loop Counter ≠ 0, go to PO
PO+5	INC	CJP	--	--	--	--	--	--	--	Loop Counter = 0, check intpt.

Table C-4
Execution Phase--Logical and I/O Instructions

Function	Address	ALU			C_n	MUX. A	MUX. B	MUX. C	Shift Control	R-Count Control	Enable Control
		S	D	F							
AND	AN	--	--	--	-	LDP	LDR	--	--	--	MP
	AN+1	AB	RM	AND	-	--	--	--	--	--	MP
OR	OR	--	--	--	-	LDP	LDR	--	--	--	MP
	OR+1	AB	RM	OR	-	--	--	--	--	--	MP
EXCLUSIVE OR	EO	--	--	--	-	--	--	--	--	--	MP
	EO+1	AB	RM	EOR	-	--	--	--	--	--	MP
LOAD ONE'S COMPLEMENT	LO	--	--	--	-	LDP	LDR	--	--	--	MP
	LO+1	0A	RM	MIN	0	--	--	--	--	--	MP
REGISTER INPUT COMMAND	RI	--	--	--	-	LDP	--	--	--	--	MP
	RI+1	0A	F	OR	-	--	--	--	--	--	MP
	RI+2	--	--	--	-	--	--	--	--	--	MP, BR

Table C-4 (Continued)
Execution Phase--Logical and I/O Instructions

Address	Jump Address	Next Address	Cond. Select	Intpt. Control	Register Control	I-Bus Control			Misc.	Remarks
						Brq	Drcv	Iosl		
AN	--	CON	--	--	--	--	--	--	--	'DO'→A, R→B
AN+1	INC	CJP	TC	--	--	--	--	--	--	(DO). AND. (R)→R, Check intpt.
OR	--	--	--	--	--	--	--	--	--	'DO'→A, R→B
OR+1	INC	CJP	TC	--	--	--	--	--	--	(DO). OR. (R)→R, Check intpt.
EO	--	CON	--	--	--	--	--	--	--	'DO'→A, R→B
EO+1	INC	CJP	TC	--	--	--	--	--	--	(DO). EOR. (R)→R, Check intpt.
LO	--	CON	--	--	--	--	--	--	--	'DO'→A, R→B
LO+1	INC	CJP	TC	--	--	--	--	--	--	$(\overline{DO})$→R, Check intpt.
RI	--	CON	--	--	--	--	--	--	--	CAW→A
RI+1	--	CON	--	--	MAI	--	--	--	--	CAW→MAR.
RI+2	--	CON	--	--	--	1	0	0	--	Request for I-Bus Control

Table C-4 (Continued)
Execution Phase--Logical and I/O Instructions

Address	Jump Address	Next Address	Cond. Select	Intpt. Control	Register Control	I-Bus Control			Misc.	Remarks
						Brq	Drcv	Iosl		
RI+3	RI+2	JRP	TCP	--	MBI	--	--	--	--	Data from Device Stored in MBR.
RI+4	INC	CJP	TC	--	--	--	--	--	--	Data ⟶ R, Check interrupt.

Table C-4 (Continued)
Execution Phase--Logical and I/O Instructions

Function	Address	ALU			C_n	MUX. A	MUX. B	MUX. C	Shift Control	R-Count Control	Enable Control
		S	D	F							
REGISTER INPUT COMMAND (Continued)	RI+3	--	--	--	-	LDR	--	--	--	--	MP
	RI+4	D0	RM	OR	-	--	--	--	--	--	MP

Table C-5
Execution Phase--Bit Manipulation Instructions

Function	Address	ALU			C_n	MUX. A	MUX. B	MUX. C	Shift Control	R-Count Control	Enable Control
		S	D	F							
REGISTER OUTPUT COMMAND	RO	--	--	--	-	LDP	--	--	--	--	MP
	RO+1	0A	F	OR	-	--	LDR	--	--	--	MP, AL
	RO+2	0B	F	OR	-	--	--	--	--	--	MP, AL
	RO+3	--	--	--	-	--	--	--	--	--	MP
	RO+4	--	--	--	-	--	--	--	--	--	MP
	RO+5										
SET BIT LOWER BYTE	SL	--	--	--	-	LDP	--	--	--	--	MP, PB
	SL+1	DA	F	AND	-	LDP	LDP	--	--	--	MP, LC, PB
	SL+2	DA	RM	OR	-	--	--	--	--	--	MP
CLEAR BIT LOWER BYTE	CL	--	--	--	-	--	--	--	--	--	MP, PB
	CL+1	DA	F	AND	-	LDP	LDP	--	--	--	MP, LC, PB

Table C-5 (Continued)
Execution Phase--Bit Manipulation Instructions

Address	Jump Address	Next Address	Cond. Select	Intpt. Control	Register Control	I-Bus Control			Misc.	Remarks
						Brq	Drcv	Iosl		
RO	--	CON	--	--	--	--	--	--	--	'CAW' → A
RO+1	--	CON	--	--	MAI	--	--	--	--	(CAW) → MAR, R → B
RO+2	--	CON	--	--	MBO	--	--	--	--	(R) → MBR
RO+3	--	CON	--	--	--	1	1	0	--	I-Bus Requested.
RO+4	RO+3	JRP	TCP	--	--	--	--	--	--	Check for transfer complete.
RO+5	INC	CJP	TC	--	--	--	--	--	--	Check interrupt.
SL	--	CON	--	--	--	--	--	--	--	EAR → A, Mask → D
SL+1	--	CON	--	--	--	--	--	--	--	Corrected 'Z' latched in CCR
SL+2	INC	CJP	TC	--	--	--	--	--	--	R^{th} bit in Lower Byte Set.
CL	--	CON	--	--	--	--	--	--	--	EAR → A, Mask → D
CL+1	--	CON	--	--	--	--	--	--	CZF	Corrected 'Z' latched in CCR

Table C-5 (Continued)
Execution Phase--Bit Manipulation Instructions

Function	Address	ALU			C_n	MUX. A	MUX. B	MUX. C	Shift Control	R-Count Control	Enable Control
		S	D	F							
CLEAR BIT LOWER BYTE (Continued)	CL+2	DA	RM	MSK	-	--	--	--	--	--	MP
TEST BIT LOWER BYTE	TL	--	--	--	-	LDP	--	--	--	--	MP, PB
	TL+1	DA	F	AND	-	--	--	--	--	--	MP, LC
SET BIT UPPER BYTE	SU	--	--	--	-	LDP	--	--	--	--	MP, PB
	SU+1	DA	F	AND	-	LDP	LDP	--	--	--	MP, LC, PB
	SU+2	DA	RM	OR	-	--	--	--	--	--	MP
CLEAR BIT UPPER BYTE	CU	--	--	--	-	LDP	--	--	--	--	MP, PB
	CU+1	DA	F	AND	-	LDP	LDP	--	--	--	MP, LC, PB
	CU+2	DA	RM	MSK	-	--	--	--	--	--	MP
TEST BIT UPPER BYTE	TU	--	--	--	-	LDP	--	--	--	--	MP, PB
	TU+1	DA	F	AND	-	--	--	--	--	--	MP, LC

Table C-5 (Continued)
Execution Phase--Bit Manipulation Instructions

Address	Jump Address	Next Address	Cond. Select	Intpt. Control	Register Control	I-Bus Control			Misc.	Remarks
						Brq	Drcv	Iosl		
CL+2	INC	CJP	TC	--	--	--	--	--	--	R^{th} bit in Lower Byte Cleared
TL	--	CON	--	--	--	--	--	--	--	EAR→A, Mask→D
TL+1	INC	CJP	TC	--	--	--	--	--	--	R^{th} bit in Lower Byte tested and CCR Set accordingly.
SU	--	CON	--	--	--	--	--	--	--	EAR→A, Mask→D
SU+1	--	CON	--	--	--	--	--	--	--	Corrected 'Z' latched in CCR
SU+2	INC	CJP	TC	--	--	--	--	--	--	R^{th} bit in Upper Byte Set.
CU	--	CON	--	--	--	--	--	--	--	EAR →A, Mask→D
CU+1	--	CON	--	--	--	--	--	--	CZF	Corrected 'Z' latched in CCR
CU+2	INC	CJP	TC	--	--	--	--	--	--	R^{th} bit in Upper Byte Set.
TU	--	CON	--	--	--	--	--	--	--	EAR→A, Mask→D
TU+1	INC	CJP	TC	--	--	--	--	--	--	R^{th} bit in Upper Byte tested and CCR Set accordingly.

Table C-6
Execution Phase--Program and Interrupt Control Instructions

Function	Address	ALU			C_n	MUX. A	MUX. B	MUX. C	Shift Control	R-Count Control	Enable Control
		S	D	F							
BRANCH ON CONDITION	BO	--	--	--	-	LDP	LDP	--	--	--	MP
	BO+1	AB	RA	PLS	1	--	--	--	--	--	MP, AL
	BO+2	--	--	--	-	LDP	--	--	--	--	MP
	BO+3	0A	F	OR	-	--	--	--	--	--	MP, AL
EXCHANGE STATUS WORD AND PROGRAM COUNTER	EX	--	--	--	-	LDP	--	--	--	--	MP
	EX+1	0A	F	OR	-	LDP	LDP	--	--	--	MP, AL
	EX+2	AB	RA	MIN	1	--	--	--	--	--	MP, AL
	EX+3	--	--	--	-	LDP	LDP	--	--	--	MP
	EX+4	AB	RA	MIN	1	--	--	--	--	--	MP, AL
	EX+5	--	--	--	-	LDP	LDP	--	--	--	MP
	EX+6	0A	F	OR	-	LDP	--	--	--	--	MP, AL
	EX+7	0A	F	PLS	1	--	--	--	--	--	MP, AL

Table C-6 (Continued)
Execution Phase--Program and Interrupt Control Instructions

Address	Jump Address	Next Address	Cond. Select	Intpt. Control	Register Control	I-Bus Control			Misc.	Remarks
						Brq	Drcv	Iosl		
BO	BO+2	CJP	BOC	--	--	--	--	--	--	If BOC = 1, go to BO+2
BO+1	INC	CJP	TC	--	MAI	--	--	--	--	PC→MAR, PC+1→PC
BO+2	--	CON	--	--	--	--	--	--	--	EAR→A
BO+3	INC	CJP	TC	--	MAI	--	--	--	--	(EAR)→MAR, Check intpt.
EX	--	CON	--	--	--	--	--	--	--	PC→A
EX+1	--	CON	--	--	MBO	--	--	--	--	(PC)→MBR, ISP→A&B
EX+2	WIM	CSP	--	--	MAI	--	--	--	--	(ISP)→MAR, (ISP)-1→ISP Call "WRITE"
EX+3	--	CON	--	--	MBO	--	--	--	--	ISP→A&B, (SW)→MBR
EX+4	WIM	CSP	--	--	MAI	--	--	--	--	(ISP)→MAR, Call "WRITE"
EX+5	--	CON	--	--	--	--	--	--	--	(EAR)→PC, EAR→A
EX+6	--	CON	--	--	--	--	--	--	--	(EAR)+1→SW, Check intpt.
EX+7	INC	CJP	TC	--	--	--	--	--	SWI	(EAR)+1→SW, Check intpt.

Table C-6 (Continued)
Execution Phase--Program and Interrupt Control Instructions

Function	Address	ALU			C_n	MUX. A	MUX. B	MUX. C	Shift Control	R-Count Control	Enable Control
		S	D	F							
RETURN FROM INTERRUPT	RI	--	--	--	-	LDP	LDP	--	--	--	MP
	RI+1	0A	F	PLS	1	--	--	--	--	--	MP, AL
	RI+2	D0	F	OR	-	LDP	LDP	--	--	--	MP, AL
	RI+3	0A	F	PLS	1	--	--	--	--	--	MP, AL
	RI+4	--	--	--	-	--	LDP	--	--	--	MP
	RI+5	D0	RM	OR	-	--	--	--	--	--	MP, AL

Table C-6 (Continued)
Execution Phase--Program and Interrupt Control Instructions

Address	Jump Address	Next Address	Cond. Select	Intpt. Control	Register Control	I-Bus Control			Misc.	Remarks
						Brq	Drcv	Iosl		
RI	--	CON	--	--	--	--	--	--	--	ISP ⟶ A&B
RI+1	RFM	CSP	TC	--	MAI	--	--	--	--	(ISP)+1 ⟶ ISP, (ISP) ⟶ MAR Call "READ"
RI+2	--	CON	--	--	--	--	--	--	--	New value loaded in SW Reg.
RI+3	RFM	CSP	TC	--	MAI	--	--	--	--	(ISP)+1 ⟶ ISP, (ISP) ⟶ MAR Call "READ"
RI+4	--	CON	--	--	--	--	--	--	--	PC ⟶ B
RI+5	INC	CJP	TC	--	--	--	--	--	--	New value loaded in PC Reg.

AD-A053 346 AIR FORCE INST OF TECH WRIGHT-PATTERSON AFB OHIO SCH--ETC F/G 9/2
EMULATION OF THE PROCESSOR FOR DISTRIBUTED PROCESSOR/MEMORY SYS--ETC(U)
DEC 77 E MUHAMMAD

UNCLASSIFIED AFIT/GE/EE/77-31 NL

3 OF 3
AD
A053346

END
DATE
FILMED
6-78
DDC

Table C-7
Execution Phase-- Arithmetic Instructions

Function	Address	ALU			C_n	MUX. A	MUX. B	MUX. C	Shift Control	R-Count Control	Enable Control
		S	D	F							
LOAD 2'S COMPLEMENT	LT	--	--	--	-	LDP	LDR	--	--	--	MP
	LT+1	0A	RM	MIN	1	--	--	--	--	--	MP, AL
ADD	AD	--	--	--	-	LDP	LDR	--	--	--	MP
	AD+1	AB	RM	PLS	0	--	--	--	--	--	MP, AL
SUBTRACT	SB	--	--	--	-	LDP	LDR	--	--	--	MP
	SB+1	AB	RM	MIN	1	--	--	--	--	--	MP, AL
COMPARE SIGNED	CS	--	--	--	-	LDP	LDR	--	--	--	MP
	CS+1	AB	RM	MIN	1	--	--	--	--	--	MP, AL

Table C-7 (Continued)
Execution Phase--Arithmetic Instructions

Address	Jump Address	Next Address	Cond. Select	Intpt. Control	Register Control	I-Bus Control			Misc.	Remarks
						Brq	Drcv	Iosl		
LT	--	CON	--	--	--	--	--	--	--	EAR → A, R → B
LT+1	INC	CJP	TC	--	--	--	--	--	--	$(\overline{EAR})+1 \rightarrow R$, Check intpt.
AD	--	CON	--	--	--	--	--	--	--	EAR → A, R → B
AD+1	INC	CJP	TC	--	--	--	--	--	--	(EAR)+(R) → R, Check Intpt.
SB	--	CON	--	--	--	--	--	--	--	EAR → A, R → B
SB+1	INC	CJP	TC	--	--	--	--	--	--	(R)-(EAR) → R, Check Intpt.
CS	--	CON	--	--	--	--	--	--	--	EAR → A, R → B
CS+1	INC	CJP	TC	--	--	--	--	--	--	(R)-(EAR) → R, S & Z Latched

Table C-7 (Continued)
Execution Phase--Arithmetic Instructions

Function	Address	ALU			C_n	MUX. A	MUX. B	MUX. C	Shift Control	R-Count Control	Enable Control
		S	D	F							
MULTIPLICATION	ML	--	--	--	-	LDR	LDP	--	--	--	MP
	ML+1	0A	RM	OR	-	--	--	--	--	--	MP, LC
	ML+2	--	--	--	-	LDC	--	--	--	IRC	MP
	ML+3	0A	Q	OR	-	--	--	--	--	--	MP, LC
	ML+4	--	--	--	-	LDC	LDP	--	--	--	MP
	ML+5	AB	F	EOR	-	LDP	LDP	--	--	--	MP
	ML+6	0A	RM	OR	-	--	--	--	--	--	MP, LC
	ML+7	--	--	--	-	--	--	--	--	--	MP
	ML+8	--	--	--	-	--	--	--	--	--	MP
	ML+9	--	--	--	-	LDP	LDP	--	--	--	MP
	ML+10	0A	RM	MIN	1	--	--	--	--	--	MP
	ML+11	--	--	--	-	LDC	--	--	--	--	MP
	ML+12	0A	Q	OR	-	--	--	--	--	--	MP, LC

Table C-7 (Continued)
Execution Phase--Arithmetic Instructions

Address	Jump Address	Next Address	Cond. Select	Intpt. Control	Register Control	I-Bus Control			Misc.	Remarks
						Brq	Drcv	Iosl		
ML	--	CON	--	--	--	--	--	--	PQC	EAR→A, R_9→B
ML+1	--	CON	--	--	--	--	--	--	--	(EAR)=Multiplicand→R_9
ML+2	ML+35	CJP	Zero	--	--	--	--	--	--	R+1→A. If (EAR)=0, go to ML+35.
ML+3	--	CON	--	--	--	--	--	--	--	(R+1)=Multiplier→Q Reg.
ML+4	ML+35	CJP	Zero	--	--	--	--	--	--	If (R+1)=0, go to ML+35.
ML+5	--	CON	--	--	--	--	--	--	PQS	Sign of Product Stored.
ML+6	--	CON	--	--	--	--	--	--	--	Check Sign of Multiplicand.
ML+7	ML+9	CJP	SIN	--	--	--	--	--	--	If -VC, go to ML+8
ML+8	ML+11	CJP	TC	--	--	--	--	--	--	If +VC, go to ML+10
ML+9	--	CON	--	--	--	--	--	--	--	EAR→A&B
ML+10	--	CON	--	--	--	--	--	--	--	2's complement of (EAR)→R_9.
ML+11	--	CON	--	--	--	--	--	--	--	R+1→A
ML+12	--	CON	--	--	--	--	--	--	--	Check sign of Multiplier.

Table C-7 (Continued)
Execution Phase--Arithmetic Instructions

Function	Address	ALU			C_n	MUX. A	MUX. B	MUX. C	Shift Control	R-Count Control	Enable Control
		S	D	F							
MULTIPLICATION (Continued)	ML+13	--	--	--	-	--	--	--	--	--	MP
	ML+14	--	--	--	-	--	--	--	--	--	MP
	ML+15	--	--	--	-	LDC	--	--	--	--	MP
	ML+16	0A	Q	MIN	1	--	--	--	--	--	MP
	ML+17	--	--	--	-	LDR	LDR	--	--	--	MP
	ML+18	0A	RM	AND	-	--	--	--	--	--	MP
	ML+19	--	--	--	-	--	--	--	--	--	MP
	ML+20	--	--	--	-	LDP	--	--	--	--	MP
	ML+21	AQ	F	AND	-	--	--	--	--	--	MP, LC
	ML+22	--	--	--	-	--	--	--	--	--	MP
	ML+23	--	--	--	-	--	--	--	--	--	MP
	ML+24	--	--	--	-	LDR	LDR	--	--	--	MP
	ML+25	AQ	RM	PLS	0	--	--	--	--	--	MP

Table C-7 (Continued)
Execution Phase--Arithmetic Instructions

Address	Jump Address	Next Address	Cond. Select	Intpt. Control	Register Control	I-Bus Control			Misc.	Remarks
						Brq	Drcv	Iosl		
ML+13	ML+15	CJP	SIN	--	--	--	--	--	--	If -VC, go to ML+15.
ML+14	ML+17	CJP	TC	--	--	--	--	--	--	If +VC, go to ML+17.
ML+15	--	CON	--	--	--	--	--	--	--	R+1⟶A
ML+16	--	CON	--	--	--	--	--	--	--	2's complement of (R+1)⟶Q
ML+17	--	CON	--	--	--	--	--	--	--	R⟶A&B
ML+18	--	CON	--	--	--	--	--	--	--	0⟶(R)
ML+19	15(DEC)	PSH	TC	--	--	--	--	--	--	Load loop counter, N = 15
ML+20	--	CON	--	--	--	--	--	--	--	R_{14}⟶A, (R_{14})=0001 (HEX)
ML+21	--	CON	--	--	--	--	--	--	--	Check LSB of Multiplier.
ML+22	ML+24	CJP	SIN	--	--	--	--	--	--	If LSB=1, go to ML+25.
ML+23	ML+26	CJP	TC	--	--	--	--	--	--	If LSB=0, go to ML+27.
ML+24	--	CON	--	--	--	--	--	--	--	R⟶A&B
ML+25	--	CON	--	--	--	--	--	--	--	(R)+(Q)⟶R

Table C-7 (Continued)
Execution Phase--Arithmetic Instructions

Function	Address	ALU			C_n	MUX. A	MUX. B	MUX. C	Shift Control	R-Count Control	Enable Control
		S	D	F							
MULTIPLICATION (Continued)	ML+26	--	--	--	-	LDR	--	--	--	--	MP
	ML+27	0A	QRR	OR	-	--	--	--	LRD	--	MP
	ML+28	--	--	--	-	--	--	--	--	--	MP
	ML+29	--	--	--	-	--	LDC	--	--	--	MP
	ML+30	0Q	RM	OR	-	--	--	--	--	--	MP
	ML+31	--	--	--	-	--	--	--	--	--	MP
	ML+32	--	--	--	-	LDR	LDR	--	--	--	MP
	ML+33	0A	RM	MIN	1	LDC	LDC	--	--	--	MP
	ML+34	0A	RM	MIN	1	--	--	--	--	--	MP
	ML+35	0A	F	AND	-	--	--	--	--	--	MP
DIVISION	DV	--	--	--	-	LDP	LDR	--	--	--	MP
	DV+1	AB	F	EOR	-	LDP	--	--	--	--	MP, LC
	DV+2	0A	F	OR	-	--	--	--	--	--	MP, LC

Table C-7 (Continued)
Execution Phase--Arithmetic Instructions

Address	Jump Address	Next Address	Cond. Select	Intpt. Control	Register Control	I-Bus Control			Misc.	Remarks
						Brq	Drcv	Iosl		
ML+26	--	CON	--	--	--	--	--	--	--	R→A
ML+27	--	CON	--	--	--	--	--	--	--	(R)&(Q) Shifted right.
ML+28	ML+20	RFC	--	--	--	--	--	--	--	If loop counter ≠ 0, go to ML+20.
ML+29	--	CON	--	--	--	--	--	--	--	R+1→B
ML+30	ML+32	CJP	SPQ	--	--	--	--	--	--	(Q)→R+1. If Product -VC, go to ML+32.
ML+31	ML+35	CJP	TC	--	--	--	--	--	--	If Product +VC, go to ML+35
ML+32	--	CON	--	--	--	--	--	--	--	R→A&B
ML+33	--	CON	--	--	--	--	--	--	--	Sign of (R) Corrected. Check intpt.
ML+34	INC	CJP	TC	--	--	--	--	--	--	Result is 0 - Check intput.
DV	--	CON	--	--	--	--	--	--	PQC	EAR→A, R→B
DV+1	--	CON	--	--	--	--	--	--	PQS	Sign of Quotient stored.
DV+2	--	CON	--	--	--	--	--	--	--	Check contents of Divisor.

Table C-7 (Continued)
Execution Phase--Arithmetic Instructions

Function	Address	ALU			C_n	MUX. A	MUX. B	MUX. C	Shift Control	R-Count Control	Enable Control
		S	D	F							
DIVISION (continued)	DV+3	--	--	--	-	LDR	--	--	--	--	MP
	DV+4	0A	F	OR	-	--	--	--	--	--	MP, LC
	DV+5	--	--	--	-	--	--	--	--	--	MP
	DV+6	--	--	--	-	--	--	--	--	--	MP
	DV+7	--	--	--	-	LDC	--	--	--	IRC	MP
	DV+8	0A	F	OR	-	--	--	--	--	--	MP, LC
	DV+9	--	--	--	-	--	--	--	--	--	MP
	DV+10	--	--	--	-	LDP	LDP	--	--	--	MP
	DV+11	0A	RM	OR	-	--	--	--	--	--	MP, LC
	DV+12	--	--	--	-	--	--	--	--	--	MP
	DV+13	--	--	--	-	--	--	--	--	--	MP
	DV+14	--	--	--	-	LDP	LDP	--	--	--	MP
	DV+15	0A	RM	MIN	1	--	--	--	--	--	MP

Table C-7 (Continued)
Execution Phase--Arithmetic Instructions

Address	Jump Address	Next Address	Cond. Select	Intpt. Control	Register Control	I-Bus Control			Misc.	Remarks
						Brq	Drcv	Iosl		
DV+3	DIV+48	CJP	Zero	--	--	--	--	--	--	If divisor = 0, go to DIV+48.
DV+4	--	CON	--	--	--	--	--	--	--	Check dividend MSH).
DV+5	DIV+7	CJP	Zero	--	--	--	--	--	--	If = 0, go to DIV+7.
DV+6	DIV+10	CJP	TC	--	--	--	--	--	--	If ≠ 0, go to DIV+10.
DV+7	--	CON	--	--	--	--	--	--	--	R+1→A. (R+1) = Dividend (LSH).
DV+8	--	CON	--	--	--	--	--	--	--	Check dividend (LSH).
DV+9	DIV+49	CJP	Zero	--	--	--	--	--	--	If = 0, go to DIV+49.
DV+10	--	CON	--	--	--	--	--	--	--	Addr. of divisor→A.
DV+11	--	CON	--	--	--	--	--	--	--	Check sign of divisor.
DV+12	DIV+14	CJP	SIN	--	--	--	--	--	--	If -VE, go to DIV+14.
DV+13	DIV+16	CJP	TC	--	--	--	--	--	--	If +VE, go to DIV+16.
DV+14	--	CON	--	--	--	--	--	--	--	Addr. of divisor→A&B.
DV+15	--	CON	--	--	--	--	--	--	--	2's complement of divisor →R_9.

Table C-7 (Continued)
Execution Phase--Arithmetic Instructions

Function	Address	ALU			C_n	MUX. A	MUX. B	MUX. C	Shift Control	R-Count Control	Enable Control
		S	D	F							
DIVISION (continued)	DV+16	--	--	--	-	LDR	LDR	--	--	--	MP
	DV+17	0A	RM	OR	-	--	--	--	--	--	MP, LC
	DV+18	--	--	--	-	--	--	--	--	--	MP
	DV+19	--	--	--	-	--	--	--	--	--	MP
	DV+20	--	--	--	-	LDR	LDR	--	--	--	MP
	DV+21	0A	RM	MIN	1	LDC	LDC	--	--	--	MP
	DV+22	0A	RM	MIN	1	--	--	--	--	--	MP
	DV+23	--	--	--	-	LDC	--	--	--	--	MP
	DV+24	0A	Q	OR	-	LDP	LDR	--	--	--	MP
	DV+25	AB	RM	MIN	1	--	--	--	--	--	MP, LC
	DV+26	--	--	--	-	--	--	--	--	--	MP
	DV+27	--	--	--	-	--	--	--	--	--	MP

Table C-7 (Continued)
Execution Phase--Arithmetic Instructions

Address	Jump Address	Next Address	Cond. Select	Intpt. Control	Register Control	I-Bus Control			Misc.	Remarks
						Brq	Drcv	Iosl		
DV+16	--	CON	--	--	--	--	--	--	--	Dividend (MSH)→A&B.
DV+17	--	CON	--	--	--	--	--	--	--	Check sign of Dividend.
DV+18	DIV+20	CJP	SIN	--	--	--	--	--	--	If -VE, go to DIV+20.
DV+19	DIV+23	CJP	TC	--	--	--	--	--	--	If +VE, go to DIV+24.
DV+20	--	CON	--	--	--	--	--	--	--	Dividend (MSH)→A&B.
DV+21	--	CON	--	--	--	--	--	--	--	2's complement of dividend (MSH)→R.
DV+22	--	CON	--	--	--	--	--	--	--	2's complement of dividend (LSH)→R+1.
DV+23	--	CON	--	--	--	--	--	--	--	Dividend (LSH)→A.
DV+24	--	CON	--	--	--	--	--	--	--	(R+1)→Q. Reg.
DV+25	--	CON	--	--	--	--	--	--	--	Dividend (MSH)-Divisor →Dividend.
DV+26	DIV+28	CJP	SIN	--	--	--	--	--	--	If result -VE, go to DIV+28.
DV+27	DIV+48	CJP	TC	--	--	--	--	--	--	If +VE, go to DIV+48.

Table C-7 (Continued)
Execution Phase--Arithmetic Instructions

Function	Address	ALU			C_n	MUX. A	MUX. B	MUX. C	Shift Control	R-Count Control	Enable Control
		S	D	F							
DIVISION (continued)	DV+28	--	--	--	-	LDR	LDR	--	--	--	MP
	DV+29	0A	QRL	OR	-	LDP	LDR	--	LLD	--	MP
	DV+30	AB	RM	PLS	0	--	--	--	--	--	MP
	DV+31	--	--	--	-	LDR	LDR	--	--	--	MP
	DV+32	0A	RM	OR	-	--	--	--	--	--	MP, LC
	DV+33	--	--	--	-	--	--	--	--	--	MP
	DV+34	--	--	--	-	LDP	--	--	--	--	MP
	DV+35	AQ	Q	OR	-	LDR	LDR	--	--	--	MP
	DV+36	0A	QRL	OR	-	LDP	LDR	--	LLD	--	MP
	DV+37	AB	RM	MIN	1	--	--	--	--	--	MP
	DV+38	--	--	--	-	LDR	LDR	--	--	--	MP
	DV+39	0A	QRL	OR	-	LDP	LDR	--	LLD	--	MP
	DV+40	AB	RM	PLS	0	--	--	--	--	--	MP

Table C-7 (Continued)
Execution Phase--Arithmetic Instructions

Address	Jump Address	Next Address	Cond. Select	Intpt. Control	Register Control	I-Bus Control			Misc.	Remarks
						Brq	Drcv	Iosl		
DV+28	--	CON	--	--	--	--	--	--	--	Dividend (MSH)⟶A.
DV+29	--	CON	--	--	--	--	--	--	--	Dividend shifted left.
DV+30	--	CON	--	--	--	--	--	--	--	Divisor added to dividend (LSH)
DV+31	15(DEC)	PSH	TC	--	--	--	--	--	--	Load loop counter. N=15.
DV+32	--	CON	--	--	--	--	--	--	--	Check sign of dividend.
DV+33	DIV+38	CJP	SIN	--	--	--	--	--	--	If -VE, go to DIV+41.
DV+34	--	CON	--	--	--	--	--	--	--	R14⟶A, (R14)=0001(HEX.)
DV+35	--	CON	--	--	--	--	--	--	--	'1' added in LSB of Q. Reg.
DV+36	--	CON	--	--	--	--	--	--	--	Dividend shifted left.
DV+37	DIV+41	CJP	TC	--	--	--	--	--	--	Divisor subtracted from dividend.
DV+38	--	CON	--	--	--	--	--	--	--	Dividend⟶A&B.
DV+39	--	CON	--	--	--	--	--	--	--	Dividend shifted left. 0⟶Quotient.
DV+40	--	CON	--	--	--	--	--	--	--	Divisor added to dividend.

Table C-7 (Continued)
Execution Phase--Arithmetic Instructions

Function	Address	ALU			C_n	MUX. A	MUX. B	MUX. C	Shift Control	R-Count Control	Enable Control
		S	D	F							
DIVISION (continued)	DV+41	--	--	--	-	--	LDC	--	--	--	MP
	DV+42	0Q	RM	OR	-	--	--	--	--	--	MP
	DV+43	--	--	--	-	--	--	--	--	--	MP
	DV+44	--	--	--	-	--	--	--	--	--	MP
	DV+45	--	--	--	-	LDC	LDC	--	--	--	MP
	DV+46	0A	RM	MIN	1	--	--	--	--	--	MP
	DV+47	--	--	--	-	--	--	--	--	--	MP
	DV+48	--	--	--	-	--	--	--	--	--	MP
	DV+49	--	--	--	-	LDC	LDC	--	--	--	MP
	DV+50	0A	RM	AND	-	--	--	--	--	--	MP
	DV+51	--	--	--	-	--	--	--	--	--	MP

Table C-7 (Continued)
Execution Phase--Arithmetic Instructions

Address	Jump Address	Next Address	Cond. Select	Intpt. Control	Register Control	I-Bus Control			Misc.	Remarks
						Brq	Drcv	Iosl		
DV+41	--	RFC	--	--	--	--	--	--	--	Repeat loop If counter ≠ 0.
DV+42	--	CON	--	--	--	--	--	--	--	Quotient→R+1.
DV+43	DIV+45	CJP	SPQ	--	--	--	--	--	--	If quotient -VE, go to DIV+45.
DV+44	DIV+51	CJP	TC	--	--	--	--	--	--	If +VE, go to DV+51.
DV+45	--	CON	--	--	--	--	--	--	--	R+1→A&B.
DV+46	--	CON	--	--	--	--	--	--	--	Sign of quotient corrected.
DV+47	DIV+51	CJP	TC	--	--	--	--	--	--	Go to DIV+51.
DV+48	DIV+51	CJP	TC	--	--	--	--	--	--	Set overflow & go to DIV+51.
DV+49	--	CON	--	--	--	--	--	--	--	Addr. of quotient→A&B.
DV+50	--	CON	--	--	--	--	--	--	--	0→Quotient.
DV+51	INC	CJP	TC	--	--	--	--	--	--	Check interrupt.

Table C-8
Execution Phase--Shift Instructions

Function	Address	ALU			C_n	MUX. A	MUX. B	MUX. C	Shift Control	R-Count Control	Enable Control
		S	D	F							
SHIFT FORMAT DECODING	SF	--	--	--	-	LDP	--	--	--	--	MP
	SF+1	0A	F	OR	-	--	--	--	--	--	MP, AL
	SF+2	--	--	--	-	--	--	--	--	--	MP
	SF+3	--	--	--	-	--	--	--	--	--	MP
LOGICAL RIGHT SHIFT SINGLE	LRS	--	--	--	-	--	LDR	--	--	--	MP
	LRS+1	0B	RR	OR	-	--	--	--	LRS	--	MP
	LRS+2	--	--	--	-	--	--	--	--	--	--
ARITHMETIC RIGHT SHIFT SINGLE	ARS	--	--	--	-	--	LDR	--	--	--	MP
	ARS+1	0B	RM	OR	-	--	LDR	--	--	--	MP
	ARS+2	0B	RR	OR	-	--	--	--	ARS	--	MP
	ARS+3	--	--	--	-	--	--	--	--	--	MP
ROTATE RIGHT SINGLE	RRS	--	--	--	-	--	LDR	--	--	--	MP

Table C-8 (Continued)
Execution Phase--Shift Instructions

Address	Jump Address	Next Address	Cond. Select	Intpt. Control	Register Control	I-Bus Control			Misc.	Remarks
						Brq	Drcv	Iosl		
SF	--	CON	--	--	--	--	--	--	--	EAR→A.
SF+1	--	CON	--	--	SBI	--	--	--	--	(EAR)→Shift Buffer Reg.
SF+2	--	CON	--	--	SBO	--	--	--	$\overline{\text{RLD}}$	Shift Count→Loop Counter.
SF+3	--	CJV	TC	--	--	--	--	--	--	Next addr. selected from PROM C.
LRS	--	CON	--	--	--	--	--	--	--	R→B.
LRS+1	LRS	RPC	--	--	--	--	--	--	--	(R) Shifted right. Repeat if counter ≠ 0.
LRS+2	INC	CJP	--	--	--	--	--	--	--	Check interrupt.
ARS	--	CON	--	--	--	--	--	--	--	R→B.
ARS+1	--	CON	--	--	--	--	--	--	--	Sign of (R)→Right shift MUX
ARS+2	ARS	RPC	--	--	--	--	--	--	--	(R) shifted right. Sign duplicated.
ARS+3	--	--	--	--	--	--	--	--	--	Check interrupt.
RRS	--	CON	--	--	--	--	--	--	--	R→B.

Table C-8 (Continued)
Execution Phase--Shift Instructions

Function	Address	ALU			C_n	MUX. A	MUX. B	MUX. C	Shift Control	R-Count Control	Enable Control
		S	D	F							
ROTATE RIGHT SINGLE (continued)	RRS+1	0B	RR	OR	-	--	--	--	RRS	--	MP
	RRS+2	--	--	--	-	--	--	--	--	--	MP
LOGICAL LEFT SHIFT SINGLE	LLS	--	--	--	-	--	LDR	--	--	--	MP
	LLS+1	0B	RL	OR	-	--	--	--	LLS	--	MP
	LLS+2	--	--	--	-	--	--	--	--	--	MP
ARITHMETIC LEFT SHIFT SINGLE	ALS	--	--	--	-	LDR	LDP	--	--	--	MP
	ALS+1	0A	RM	OR	-	--	LDR	--	--	--	MP
	ALS+2	0B	RL	OR	-	LDP	LDR	--	LLS	--	MP
	ALS+3	AB	F	EOR	-	--	--	--	--	--	MP
	ALS+4	--	--	--	-	--	--	--	--	--	MP
	ALS+5	--	--	--	-	--	--	--	--	--	MP
ROTATE LEFT SINGLE	RLS	--	--	--	-	--	LDR	--	--	--	MP

Table C-8 (Continued)
Execution Phase--Shift Instructions

Address	Jump Address	Next Address	Cond. Select	Intpt. Control	Register Control	I-Bus Control			Misc.	Remarks
						Brq	Drcv	Iosl		
RRS+1	RRS	RPC	--	--	--	--	--	--	--	(R)rotated right. Repeat if counter ≠ 0.
RRS+2	INC	CJP	--	--	--	--	--	--	--	Check interrupt.
LLS	--	CON	--	--	--	--	--	--	--	R⟶B.
LLS+1	LLS	RPC	--	--	--	--	--	--	--	(R) Shifted left. Repeat if counter ≠ 0.
LLS+2	INC	CJP	--	--	--	--	--	--	--	Check interrupt.
ALS	--	CON	--	--	--	--	--	--	--	R⟶A, R_{15}⟶B.
ALS+1	--	CON	--	--	--	--	--	--	--	(R)⟶R_{15}, R⟶B.
ALS+2	ALS	RPC	--	--	--	--	--	--	--	(R) Shifted left. Repeat if counter ≠ 0.
ALS+3	ALS+5	CJP	SIN	--	--	--	--	--	--	(R)·EOR·(R_{15})·Check result.
ALS+4	INC	CJP	TC	--	--	--	--	--	--	Result +VE, check intpt.
ALS+5	INC	CJP	TC	--	--	--	--	--	SOF	Result -VE, set OVERFLOW.
RLS	--	CON	--	--	--	--	--	--	--	R⟶B.

Table C-8 (Continued)
Execution Phase--Shift Instructions

Function	Address	ALU			C_n	MUX. A	MUX. B	MUX. C	Shift Control	R-Count Control	Enable Control
		S	D	F							
ROTATE LEFT SINGLE (continued)	RLS+1	0B	RL	OR	-	--	--	--	RLS	--	MP
	RLS+2	--	--	--	-	--	--	--	--	--	MP
LOGICAL RIGHT SHIFT DOUBLE	LRD	--	--	--	-	--	LDC	--	--	IRC	MP
	LRD+1	0B	Q	OR	-	LDR	--	--	--	--	MP
	LRD+2	0B	QRR	OR	-	--	--	--	LRD	--	MP
	LRD+3	--	--	--	-	--	--	--	--	--	MP
ARITHMETIC RIGHT SHIFT DOUBLE	ARD	--	--	--	-	--	LDC	--	--	IRC	MP
	ARD+1	0B	Q	OR	-	LDR	--	--	--	--	MP
	ARD+2	0A	RM	OR	-	--	LDR	--	--	--	MP
	ARD+3	0B	QRR	OR	-	--	LDP	--	ARD	--	MP
	ARD+4	0Q	RM	OR	-	LDR	LDP	--	--	--	MP
	ARD+5	AB	RM	AND	-	LDP	LDC	--	--	--	MP
	ARD+6	AB	RM	AND	-	--	--	--	--	--	MP

Table C-8 (Continued)
Execution Phase--Shift Instructions

Address	Jump Address	Next Address	Cond. Select	Intpt. Control	Register Control	I-Bus Control			Misc.	Remarks
						Brq	Drcv	Iosl		
RLS+1	RLS	RPC	--	--	--	--	--	--	--	(R) rotated left. Repeat if counter ≠ 0.
RLS+2	INC	CJP	--	--	--	--	--	--	--	Check interrupt.
LRD	--	CON	--	--	--	--	--	--	--	R+1⟶B.
LRD+1	--	CON	--	--	--	--	--	--	--	(R+1)⟶Q Reg., R⟶B.
LRD+2	LRD	RPC	--	--	--	--	--	--	--	(R)&(Q) Shifted right (Logic).
LRD+3	INC	CJP	--	--	--	--	--	--	--	Check interrupt.
ARD	--	CON	--	--	--	--	--	--	--	R+1⟶B.
ARD+1	--	CON	--	--	--	--	--	--	--	(R+1)⟶Q.Reg., R⟶A.
ARD+2	--	CON	--	--	--	--	--	--	--	Sign of (R)⟶Right Shift MUX.
ARD+3	ARD	RPC	--	--	--	--	--	--	--	(R)&(Q) Shifted right (Arith.)
ARD+4	--	CON	--	--	--	--	--	--	--	(Q)⟶R+1, R⟶A, R5⟶B.
ARD+5	--	CON	--	--	--	--	--	--	--	Sign of (R5) Set to Sign of (R)
ARD+6	INC	CJP	--	--	--	--	--	--	--	Sign of (R+1) Set to Sign of (R5) Check interrupt.

Table C-8 (Continued)
Execution Phase--Shift Instructions

Function	Address	ALU			C_n	MUX. A	MUX. B	MUX. C	Shift Control	R-Count Control	Enable Control
		S	D	F							
ROTATE RIGHT DOUBLE	RRD	--	--	--	-	--	LDC	--	--	IRC	MP
	RRD+1	0B	Q	OR	-	--	LDR	--	--	--	MP
	RRD+2	0B	QRR	OR	-	--	LDP	--	RRD	--	MP
	RRD+3	0Q	RM	OR	-	--	--	--	--	--	MP
LOGICAL SHIFT LEFT DOUBLE	LLD	--	--	--	-	--	LDC	--	--	IRC	MP
	LLD+1	0B	Q	OR	-	--	LDR	--	--	--	MP
	LLD+2	0B	QRL	OR	-	--	LDP	--	LLD	--	MP
	LLD+3	0Q	RM	OR	-	--	--	--	--	--	MP
ROTATE LEFT DOUBLE	RLD	--	--	--	-	--	LDC	--	--	IRC	MP
	RLD+1	0B	Q	OR	-	--	LDR	--	--	--	MP
	RLD+2	0B	QRL	OR	-	--	--	--	RLD	--	MP
	RLD+3	--	--	--	-	--	--	--	--	--	MP

Table C-8 (Continued)
Execution Phase--Shift Instructions

Address	Jump Address	Next Address	Cond. Select	Intpt. Control	Register Control	i-Bus Control			Misc.	Remarks
						Brq	Drcv	Iosl		
RRD	--	CON	--	--	--	--	--	--	--	R+1⟶B.
RRD+1	--	CON	--	--	--	--	--	--	--	(R+1)⟶Q. Reg. R⟶B.
RRD+2	RRD	RPC	--	--	--	--	--	--	--	(R)&(Q) rotated right.
RRD+3	INC	CJP	--	--	--	--	--	--	--	Count = 0, Check intpt.
LLD	--	CON	--	--	--	--	--	--	--	R+1⟶B.
LLD+1	--	CON	--	--	--	--	--	--	--	(R+1)⟶Q, R⟶B.
LLD+2	LLD	RPC	--	--	--	--	--	--	--	(R)&(Q) Shifted left. Check count.
LLD+3	INC	CJP	--	--	--	--	--	--	--	(Q)⟶R+1, Check Intpt.
RLD	--	CON	--	--	--	--	--	--	--	R+1⟶B.
RLD+1	--	CON	--	--	--	--	--	--	--	(R+1)⟶Q. Reg., R⟶B.
RLD+2	RLD	RPC	--	--	--	--	--	--	--	(R)&(Q) Rotated left.
RLD+3	INC	CJP	--	--	--	--	--	--	--	Check interrupt.

Table C-9
Execution Phase--Extended Short Format Instructions

Function	Address	ALU			C_n	MUX. A	MUX. B	MUX. C	Shift Control	R-Count Control	Enable Control
		S	D	F							
LOAD DIRECT SHORT	LS	--	--	--	-	LDT	LDP	--	--	--	MP
	LS+1	DA	RM	PLS	0	LDP	LDR	--	--	--	MP, AL
	LS+2	0A	RM	OR	-	--	--	--	--	--	MP, AL
LOAD CONSTANT SHORT	LC	--	--	--	-	--	LDR	--	--	--	MP
	LC+1	D0	RM	OR	-	--	--	--	--	--	MP, AL
STORE DIRECT SHORT	SD	--	--	--	-	LDT	--	--	--	--	MP
	SD+1	DA	F	PLS	0	--	LDR	--	--	--	MP, AL
	SD+2	0B	F	OR	-	--	--	--	--	--	MP, AL
	SD+3	--	--	--	-	--	--	--	--	--	MP
ADD CONSTANT SHORT	AC	--	--	--	-	LDR	LDR	--	--	--	MP
	AC+1	DA	RM	PLS	0	--	--	--	--	--	MP, AL

Table C-9 (Continued)
Execution Phase--Extended Short Format Instructions

Address	Jump Address	Next Address	Cond. Select	Intpt. Control	Register Control	I-Bus Control			Misc.	Remarks
						Brq	Drcv	Iosl		
LS	--	CON	--	--	SIO	--	--	--	--	T→A, EAR→B, (I)→D.
LS+1	--	CON	--	--	--	--	--	--	--	(T)+(I)→EAR, R→B.
LS+2	INC	CJP	TC	--	--	--	--	--	--	(EAR)→R, Check intpt.
LC	--	CON	--	--	SIO	--	--	--	--	R→B, I (Sign extended) →D.
LC+1	INC	CJP	TC	--	--	--	--	--	--	(I)→R, Check interrupt.
SD	--	CON	--	--	SIO	--	--	--	--	T→A, I (Sign extended) →D.
SD+1	--	CON	--	--	MAI	--	--	--	--	(T)+(I)→MAR, R→B.
SD+2	WIM	CSP	TC	--	MBO	--	--	--	--	(R)→MBR, Call "WRITE."
SD+3	INC	CJP	TC	--	--	--	--	--	--	Check Interrupt.
AC	--	CON	--	--	SIO	--	--	--	--	R→A&B, I (Sign extended →D.
AC+1	INC	CJP	TC	--	--	--	--	--	--	(I)+(R)→R, Check Intpt.

Table C-9 (Continued)
Execution Phase--Extended Short Format Instructions

Function	Address	ALU			C_n	MUX. A	MUX. B	MUX. C	Shift Control	R-Count Control	Enable Control
		S	D	F							
COMPARE CONSTANT SHORT	CS	--	--	--	-	LDR	LDR	--	--	--	MP
	CS+1	DA	RM	MIN	0	--	--	--	--	--	MP, AL
BRANCH ON CONDITION SHORT	BS	--	--	--	-	LDP	LDP	--	--	--	MP
	BS+1	AB	RA	PLS	1	--	--	--	--	--	MP, AL
	BS+2	DA	RM	PLS	0	--	--	--	--	--	MP
BRANCH INDIRECT AND LINK REGISTER	BI	--	--	--	-	LDP	LDR	--	--	--	MP
	BI+1	0A	RM	OR	-	--	LDP	--	--	--	MP
	BI+2	D0	RM	OR	-	--	--	--	--	--	MP
INCREMENT AND BRANCH IF NEGATIVE SHORT	IB	--	--	--	-	--	LDP	--	--	--	MP
	IB+1	0B	RM	PLS	1	--	--	--	--	--	MP
	IB+2	--	--	--	-	--	LDP	LDP	--	--	MP
	IB+3	--	--	--	-	--	--	--	--	--	MP
	IB+4	DA	RM	PLS	0	--	--	--	--	--	MP

Table C-9 (Continued)
Execution Phase--Extended Short Format Instructions

Address	Jump Address	Next Address	Cond. Select	Intpt. Control	Register Control	I-Bus Control			Misc.	Remarks
						Brq	Drcv	Iosl		
CS	--	CON	--	--	SIO	--	--	--	--	R⟶A&B, I(Sign extended) ⟶D.
CS+1	INC	CJP	TC	--	--	--	--	--	--	(I) compared with (R) and CCR Set accordingly.
BS	BS+2	CJP	BOC	--	SIO	--	--	--	--	PC⟶A&B, (I)⟶D.
BS+1	INC	CJP	TC	--	--	--	--	--	--	BOC≠1, Check Interrupt.
BS+2	INC	CJP	TC	--	--	--	--	--	--	BOC=1, (I)+(PC)⟶PC.
BI	--	CON	--	--	--	--	--	--	--	PC⟶A, R⟶B.
BI+1	--	CON	--	--	SIO	--	--	--	--	(PC)⟶R, PC⟶B, (I)⟶D.
BI+2	INC	CJP	TC	--	--	--	--	--	--	(I)⟶PC, Check Interrupt.
IB	--	CON	--	--	--	--	--	--	--	R⟶B.
IB+1	--	CON	--	--	--	--	--	--	--	(R)+1⟶R.
IB+2	IB+4	CJP	SIN	--	SIO	--	--	--	--	PC⟶A&B, (I)⟶D.
IB+3	INC	CJP	TC	--	--	--	--	--	--	'SIN'=0, Check Interrupt.
IB+4	INC	CJP	TC	--	--	--	--	--	--	'SIN'=1, (I)+(PC)⟶PC Check Interrupt.

Table C-10
Interrupt Handling Phase

Function	Address	ALU			C_n	MUX. A	MUX. B	MUX. C	Shift Control	R-Count Control	Enable Control
		S	D	F							
INTERRUPT CHECKING	INC	--	--	--	-	--	--	--	--	--	MP
	INC+1	--	--	--	-	--	--	--	--	--	MP
	INC+2	--	--	--	-	--	--	--	--	--	MP
INTERRUPT HANDLING	IH	--	--	--	-	--	--	--	--	--	MP
	IH+1	--	--	--	-	--	--	--	--	--	MP
	IH+2	--	--	--	-	--	LDP	--	--	--	MP
	IH+3	D0	RM	OR	-	LDP	--	--	--	--	MP
	IH+4	0A	F	OR	-	LDP	LDP	--	--	--	MP, AL
	IH+5	0B	RA	MIN	0	--	--	--	--	--	MP, AL
	IH+6	--	--	--	-	LDP	LDP	--	--	--	MP
	IH+7	0B	RA	MIN	0	--	--	--	--	--	MP, AL

Table C-10 (Continued)
Interrupt Handling Phase

Address	Jump Address	Next Address	Cond. Select	Intpt. Control	Register Control	I-Bus Control			Misc.	Remarks
						Brq	Drcv	Iosl		
INC	IH	CJP	INI	EIR	--	--	--	--	--	If Internal Intpt., go to (IH).
INC+1	IH	CJP	INE	--	--	--	--	--	--	If External Intpt., go to (IH+1).
INC+2	FH	CJP	TC	--	--	--	--	--	--	If no Intpt., start "Fetch Phase."
IH	IH+2	CJP	TC	RVC	--	--	--	--	--	Internal Trap Vector (TV) ⟶D Bus.
IH+1	--	CON	--	--	MBI	--	--	--	--	External TV⟶D Bus, 1⟶ACK.
IH+2	--	CON	--	RSR	--	--	--	--	--	R_{13}⟶B. Clear Interrupt.
IH+3	--	CON	--	CAI	--	--	--	--	--	(TV)⟶R_{13}, PC⟶A.
IH+4	--	CON	--	--	MBO	--	--	--	--	(PC)⟶MBR, ISP⟶A&B.
IH+5	WIM	CSP	TC	--	MAI	--	--	--	--	(ISP)⟶MAR. Call "WRITE."
IH+6	--	CON	--	--	MBO	--	--	--	--	Stat. Word⟶MBR. ISP⟶A&B.
IH+7	WIM	CSP	TC	--	MAI	--	--	--	--	(ISP)⟶MAR. Call "WRITE"

Table C-10 (Continued)
Interrupt Handling Phase

Function	Address	ALU			C_n	MUX. A	MUX. B	MUX. C	Shift Control	R-Count Control	Enable Control
		S	D	F							
INTERRUPT HANDLING (continued)	IH+8	--	--	--	-	LDP	LDP	--	--	--	MP
	IH+9	0B	RA	PLS	1	--	--	--	--	--	MP, AL
	IH+10	--	--	--	-	--	LDP	--	--	--	MP
	IH+11	D0	RM	OR	-	LDP	--	--	--	--	MP
	IH+12	0A	F	OR	-	--	--	--	--	--	MP, AL
	IH+13	--	--	--	-	--	LDP	--	--	--	MP
	IH+14	D0	RM	OR	-	--	--	--	--	--	MP, AL
	IH+15	--	--	--	-	LDP	LDP	--	--	--	MP
	IH+16	AB	F	AND	-	LDP	--	--	--	--	MP, AL
	IH+17	0A	F	F	-	--	--	--	--	--	MP, AL
	IH+18	--	--	--	-	--	--	--	--	--	MP

Table C-10 (Continued)
Interrupt Handling Phase

Address	Jump Address	Next Address	Cond. Select	Intpt. Control	Register Control	I-Bus Control			Misc.	Remarks
						Brq	Drcv	Iosl		
IH+8	--	CON	--	--	--	--	--	--	--	R_{13} → A&B.
IH+9	RFM	CSP	TC	--	MAI	--	--	--	--	TV → MAR, TV+1 → R_{13}, Call "READ."
IH+10	--	CON	--	--	--	--	--	--	--	PC → B.
IH+11	--	CON	--	--	MBI	--	--	--	--	New value loaded in PC· R_{13} → A.
IH+12	RFM	CSP	TC	--	MAI	--	--	--	--	TV+1 → MAR, Call "READ"
IH+13	--	CON	--	--	--	--	--	--	--	R_{12} → B.
IH+14	--	CON	--	--	MBI	--	--	--	SWI	New Stat. Word → SW Reg. & R_{12}.
IH+15	--	CON	--	LMR	--	--	--	--	--	R_{12} → A, R_{15} → B (R_{15}) = 03FF.
IH+16	--	CON	--	--	MBO	--	--	--	--	10LSB of Stat. Word → MBR R_{10} → A.
IH+17	--	CON	--	--	MAI	--	--	--	--	(R_{10})=CAW 0000 → MAR.
IH+18	--	CON	--	--	--	1	0	0	--	Request for I-Bus.

Table C-10 (Continued)
Interrupt Handling Phase

Function	Address	ALU			C_n	MUX. A	MUX. B	MUX. C	Shift Control	R-Count Control	Enable Control
		S	D	F							
INTERRUPT HANDLING (continued)	IH+19	--	--	--	-	--	--	--	--	--	MP
	IH+20	--	--	--	-	--	--	--	--	--	MP

Table C-10 (Continued)
Interrupt Handling Phase

Address	Jump Address	Next Address	Cond. Select	Intpt. Control	Register Control	I-Bus Control			Misc.	Remarks
						Brq	Drcv	Iosl		
IH+19	IH+20	JRP	TCP	--	--	--	--	--	--	10 LSB⟶BIU.
IH+20	FH	CJP	TC	--	--	--	--	--	--	Go to "Fetch" Phase.

Table C-11
Write In MEM and Read from MEM Subroutine

Function	Address	ALU			C_n	MUX. A	MUX. B	MUX. C	Shift Control	R-Count Control	Enable Control
		S	D	F							
WRITE IN MEMORY SUBROUTINE	WIM	--	--	--	-	--	--	--	--	--	MP
	WIM+1	--	--	--	-	--	--	--	--	--	MP
	WIM+2	--	--	--	-	--	--	--	--	--	MP
READ FROM MEMORY SUBROUTINE	RFM	--	--	--	-	--	--	--	--	--	MP
	RFM+1	--	--	--	-	--	--	--	--	--	MP, BR
	RFM+2	--	--	--	-	--	--	--	--	--	MP

Table C-11 (Continued)
Write In MEM and Read from MEM Subroutine

Address	Jump Address	Next Address	Cond. Select	Intpt. Control	Register Control	I-Bus Control			Misc.	Remarks
						Brq	Drcv	Iosl		
WIM	--	CON	--	--	--	1	0	1	--	Request for I-Bus Control.
WIM+1	WIM+2	JRP	TCP	--	--	--	--	--	--	Check for "Transfer Complete."
WIM+2	--	RTN	TC	--	--	--	--	--	--	Return from Subroutine.
RFM	--	CON	--	--	--	1	1	1	--	Request for I-Bus Control.
RFM+1	RFM+2	JRP	TCP	--	--	--	--	--	--	Check for "Transfer Complete."
RFM+2	--	RTN	TC	--	MBI	--	--	--	--	(MEM.)→MBR. Return from Subroutine.

Appendix D

Control Fields

This appendix contains the tables of the Control Fields as applicable to the microinstruction format. These tables are already given at appropriate places in Chapter III and Chapter IV. They have, however, been regrouped in this appendix for ease of reference.

Table D-1
ALU Source Control Field

Micro Code			ALU Source Operands		Mnemonic
I_2	I_1	I_0	R	S	
0	0	0	A	Q	AQ
0	0	1	A	B	AB
0	1	0	0	Q	0Q
0	1	1	0	B	0B
1	0	0	0	A	0A
1	0	1	D	A	DA
1	1	0	D	Q	DQ
1	1	1	D	0	D0

Table D-2
ALU Function Control Field

Micro Code			ALU Function	Mnemonic
I_5	I_4	I_3		
0	0	0	R Plus S	PLS
0	0	1	S Minus R	MIN
0	1	0	R Minus S	MIN
0	1	0	R Or S	OR
1	0	0	R and S	AND
1	0	1	$\overline{R}$ and S	MSK
1	1	0	R Ex-Or S	EOR
1	1	1	R Ex-Nor S	ENR

Table D-3
ALU Destination Control Field

Micro Code I_8	I_7	I_6	Y-Output	Mnemonic	Explanation
0	0	0	F	Q	Result appears at "Y" and also stored in Q.
0	0	1	F	F	Result appears at "Y".
0	1	0	A	RA	A-port data appears at "Y," result stored in RAM.
0	1	1	F	RM	Result stored in RAM.
1	0	0	F	QRR	Result shifted right and stored in RAM and Q.
1	0	1	F	RR	Result shifted right and stored in RAM.
1	1	0	F	QRL	Result shifted left and stored in RAM and Q.
1	1	1	F	RL	Result shifted left and stored in RAM.

Table D-4
Control Fields for Multiplexers A and B

Micro Code I_1	I_0	Mnemonic	Explanation
0	0	LDP	Select 4-bits from P. L. Reg.
0	1	LDT	Select 4-bits from T-field.
1	0	LDR	Select 4-bits from R-field.
1	1	LDC	Select 4-bits from R-counter output.

Table D-5
Control Field for Multiplexer C

Micro Code I_1	I_0	Mnemonic	Explanation
0	0	x x	x x x
0	1	AM	Select Addressing Mode
1	0	OC1	Select OP-Code C1
1	1	OC2	Select OP-Code C2

Table D-6
Shift Control Field

Micro Code S_3	S_2	S_1	S_0	I_8	I_7	I_6	Mnemonic	Explanation
1	1	0	0	1	0	1	LRS	Logical shift right single
1	1	0	0	1	0	0	LRD	Logical shift right double
1	0	0	0	1	0	1	ARS	Arithmetic shift right single
1	0	0	0	1	0	0	ARD	Arithmetic shift right double
0	1	0	0	1	0	1	RRS	Rotate right single
0	0	0	0	1	0	0	RRD	Rotate right double
0	0	0	0	1	1	0	LLD	Logical shift left double
0	0	1	1	1	1	0	LLS	Logical shift left single
0	0	1	1	1	1	1	ALS	Arithmetic shift left single
0	0	0	1	1	1	1	RLS	Rotate left single
0	1	0	0	1	1	0	RLD	Rotate left double

Table D-7
R-Counter Control Field

Micro Code I_1	Micro Code I_0	Mnemonic	Explanation
0	1	DCR	Decrement R-Counter
1	1	ICR	Increment R-Counter
0	0	x	x x
1	0	x	x x

Table D-8
"Enable" Control Field

Micro Codes I_6	I_5	I_4	I_3	I_2	I_1	I_0	Mnemonic	Explanation
0	0	0	0	0	0	1	MP	Enable output of microprogram controller.
0	0	0	0	0	1	0	AL	Enable output of ALU.
0	0	0	0	1	0	0	ET	Enable T-field at loop counter I/P.
0	0	0	1	0	0	0	PB	Enable PROM B.
0	0	1	0	0	0	0	LC	Load Condition Code Register.
0	1	0	0	0	0	0	BR	Load data in MBR.
1	0	0	0	0	0	0	IC	Load data in CIR.

Table D-9
Control Instructions for Microprogram Controller
(Next Address Control Field)

Micro Code I_3	I_2	I_1	I_0	Mnemonic	Instruction	Enable
0	0	0	0	JZ	Jump to Address Zero.	PL
0	0	0	1	CSP	Cond. jump to subroutine; address in P. L. Reg.	PL
0	0	1	0	JMA	Jump to address at MAP. PROM output.	MAP
0	0	1	1	CJP	Cond. jump to address in P. L. Reg.	PL
0	1	0	0	PSH	Push stack and conditionally load counter.	PL
0	1	0	1	SRP	Cond. jump to subroutine; ADDRESS IN "R"/P. L. Reg.	PL
0	1	1	0	CJV	Cond. jump to vector address.	VEC
0	1	1	1	JRP	Cond. jump to address in "R"/P. L. Reg.	PL
1	0	0	0	RFC	Repeat loop if counter ≠ 0.	PL
1	0	0	1	RPC	Repeat P. L. Reg. address if counter ≠ 0.	PL
1	0	1	0	RTN	Cond. return from subroutine.	PL
1	0	1	1	JPP	Cond. jump to P. L. Address and pop stack.	PL
1	1	0	0	LCC	Load counter and continue.	PL
1	1	0	1	LP	Test end of loop.	PL
1	1	1	0	CON	Continue.	PL
1	1	1	1	TWB	Three-way branch.	PL

Table D-10
Condition Selection Control Field

Micro Code I_3	I_2	I_1	I_0	Mnemonic	Condition Selected
0	0	0	0	INI	Internal Interrupt
0	0	0	1	INE	External Interrupt
0	0	1	0	AO	Derived Address/Derived Operand
0	0	1	1	OVF	Overflow
0	1	0	0	Z	Zero
0	1	0	1	SIN	Sign
0	1	1	0	TCP	Transfer Complete
0	1	1	1	T = 7	T = 7
1	0	0	0	MIO	Memory Input/Output
1	0	0	1	TC	True Condition
1	0	1	0	HLT	Halt
1	0	1	1	BOC	Branch-on One of 8-Conditions
1	1	0	0	IOC	Illegal OP-Code
1	1	0	1	x	x
1	1	1	0	x	x
1	1	1	1	x	x

Table D-11
Interrupt Control Field

Micro Code I_3	I_2	I_1	I_0	Mnemonic	Explanation
0	0	0	0	MCL	Master Clear.
0	0	0	1	CAI	Clear All Interrupts.
0	1	0	0	CIV	Clear Interrupt, Last Vector Read
0	1	0	1	RVC	Read Vector.
0	1	1	0	RSR	Read Status Register.
1	1	0	0	CMR	Clear Mask Register.
1	1	0	1	DIR	Disable Interrupt Request.
1	1	1	0	LMR	Load Mask Register.
1	1	1	1	EIR	Enable Interrupt Request.

Table D-12
Register Control Field

Micro Code I_2	I_1	I_0	Mnemonic	Explanation
0	0	0	xx	x x
0	0	1	MAI	Load Address in MAR.
0	1	0	MBI	Put data on D-Bus (from MBR).
0	1	1	MBO	Put data in MBR for transmission.
1	0	0	LIR	Load I-Field in I-Register.
1	0	1	SIO	Put sign extended I-Field on D-Bus.
1	1	0	SBI	Load data into Shift Buffer Register.
1	1	1	SBO	Put data out from Shift Buff. Register

Table D-13
"Miscellaneous" Control Field

Micro Code I_3 I_2 I_1 I_0	Mnemonic	Explanation
0 0 0 0	x x	x x
0 0 0 1	CZF	Correct "Z"-flag.
0 0 1 0	$\overline{\text{RLD}}$	Load Loop Counter.
0 0 1 1	SOV	Set Over Flow flag.
0 1 0 0	GAK	Generate Acknowledge.
0 1 0 1	LVE	Load Vector.
0 1 1 0	SWI	Load Status Word in Status Word Reg.
0 1 1 1	SWO	Enable Output of Status Word Reg.
1 0 0 0	CSW	Clear Status Word.
1 0 0 1	SPQ	Set Product/Quotient Flip Flop.
1 0 1 0	CPQ	Clear Product/Quotient Flip Flop.
All Others	x x	x x

Vita

Ejaz Muhammad was born April 3, 1946 at Ajmer, India. He attended Islamia High School Khazana Gate at Lahore, getting his matriculation in 1962. He then attended Government College at Lahore and obtained his FSC in 1964.

In 1966, he joined the Pakistan Air Force as a cadet. He attended the Pakistan Air Force College of Aeronautical Engineering at Karachi, where he graduated in 1969 with a B.E. degree in Avionics Engineering. Upon graduation he was commissioned as a Flying Officer in the engineering branch.

He served on various duties at different maintenance assignments in PAF before being selected for the resident Graduate Electrical Engineering program at AFIT. He graduated with an M.S. degree in Electrical Engineering on 16 December 1977.

Permanent address: c/o P.A.F. Officers' Mess
Koranzi Creak
Karachi
Pakistan

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER AFIT/GE/EE/77-31	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) EMULATION OF THE PROCESSOR FOR DISTRIBUTED PROCESSOR/MEMORY SYSTEM USING Am 2900 MICROPROCESSOR CHIP SET		5. TYPE OF REPORT & PERIOD COVERED MS Thesis 6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) Ejaz Muhammad Flt Lt, PAF		8. CONTRACT OR GRANT NUMBER(s)
9. PERFORMING ORGANIZATION NAME AND ADDRESS Air Force Institute of Technology (AFIT/EN) Wright-Patterson AFB, Ohio 45433		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS Air Force Avionics Laboratory (AFAL/AAT) Wright-Patterson AFB, Ohio 45433		12. REPORT DATE December 1977 13. NUMBER OF PAGES 237
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office)		15. SECURITY CLASS. (of this report) UNCLASSIFIED 15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

(Master's Thesis)

18. SUPPLEMENTARY NOTES

Approved for public release; IAW AFR 190-17

JERRAL F. GUESS, Captain, USAF
Director of Information

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Processor Emulator
Processor for DP/M System
Processor design using Am 2900 chip set

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

The processor for the Distributed Processor (DP/M) System is a 16-bit, two's complement, fixed point, eight register file architecture. This processor was designed using Am 2900 microprocessor chip set. The design used four CPU chips (Am 2901), one microprogram controller chip (Am 2910), and eight PROM chips (Intel 3604A-2) to implement the microprogram memory. A number of multiplexers, registers, tri-state buffers, and

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

counters were utilized to augment the basic design. A micro-level "Monitor" was also implemented. A 64-bit microinstruction word format was determined to provide the control signals for the processor hardware. Flow charts were drawn and micro-codes were tabulated for the specified instruction set. The flow charts and the micro-codes were arranged to conform to the state transition diagram of the processor. The Am 2900 microprocessor chip set was found to be a very powerful and flexible source for emulating the DP/M System. The design presented in this report can be hardwired to realize a Lab model of the DP/M Processor.